VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

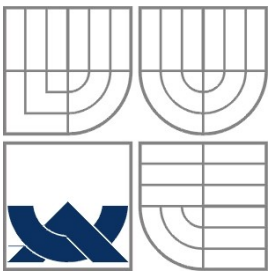
3D PROJEKCE FOTEK V OPENGL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

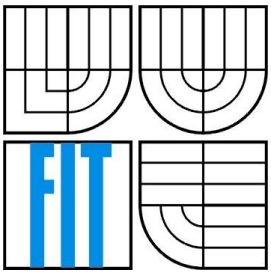
AUTOR PRÁCE
AUTHOR

BC. MARTIN DOKOUPIL

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

3D PROJEKCE FOTEK V OPENGL

3D PHOTO SLIDESHOW USING OPENGL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. MARTIN DOKOUPIL

VEDOUCÍ PRÁCE
SUPERVISOR

ING. SEEMAN MICHAL

BRNO 2010

Abstrakt

Tato zpráva popisuje implementaci editoru pro projekci fotografií. Tento editor může sloužit k tvorbě dynamických sekvencí složených z fotografií. Obraz z kamery, který je dvourozměrný je nanesen na kouli v trojrozměrném prostoru a následně zobrazován. Poté je možné pohybovat kamerou po snímku a také ji přibližovat. Použitím tohoto editoru lze docílit reálněji působících scén zejména při pohybu. Při použití panoramatických snímků umožňuje editor volné rozhlížení po tomto snímku. Dokument popisuje způsob převodu dvourozměrného snímku do prostoru a také ukazuje využití transformací pro pohyb kamery a metody pro implementaci přibližování kamery. Obsahuje popis tvorby uživatelského rozhraní a také zmiňuje další rozšíření, které byly implementovány v editoru.

Abstract

This document describes implementation of photo slideshow editor. This editor can be used to create dynamic sequences from taken photos. Image taken by camera is projected onto three-dimensional sphere and drawn to screen. Then it is posible to rotate and zoom camera over image. Visual output of this editor makes moving over image more real. If panoramic image is used, user is able to free look around. This document describes method which can translate two-dimensional coordinates to three-dimensional space. Shows how to use transformations for moving camera and methods to implement camera zoom. Contains description of user interface and also mentions extensions that were implemented in editor.

Klíčová slova

editor, fotoaparát, objektiv, obrazové sekvence, kamera, zoom

Keywords

editor, camera, lens, slideshow, pan, zoom

Citace

Dokoupil, M.: 3D projekce fotek v OpenGL, diplomová práce, Brno, FIT VUT v Brně, 2010

3D Projekce fotek v OpenGL

Prohlášení

Prohlašuji, že jsem tento semestrální projekt vypracoval samostatně pod vedením Ing. Michala Seemana.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Dokoupil
24.05.2010

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Michalu Seemanovi za jeho rady a pomoc při tvorbě této práce. Zejména bych chtěl poděkovat za poskytnutí jeho knihovny pro transformaci souřadnic.

Obsah

1 Úvod

Počítačová grafika patří k rychle se rozvíjejícím odvětvím informačních technologií. Cílem rozvoje je zejména zvyšování kvality výsledného obrazu. V současnosti jsou možnosti zobrazování na velmi vysoké úrovni. Proto se může vývoj zaměřit také na přístup k tvorbě uživatelských rozhraní a způsob zobrazování grafického výstupu bez nutností být omezován výkonem počítačů. Editory pro tvorbu obrazových sekvencí již nemusí být čistě dvourozměrné. Můžeme využít dnešních možností a dvourozměrnou fotografii transformovat tak, že se promítne do trojrozměrného prostoru. Současně při tomto zobrazení lze provést korekci optických vad způsobených optickou soustavou fotoaparátu. Tím můžeme získat reálnější scénu a to zejména při rotaci pozorovatele po snímku. Toho lze nejlépe využít u panoramatických snímků, které uživateli editoru umožňují volné rozhlížení po krajině. Existují také prohlížeče panoramatických snímků, ty jsou však určeny pouze pro zobrazení scény a neumožňují tvorbu animací.

Tato práce se zabývá vytvořením právě takového editoru. Cílem je vytvořit takový nástroj, který umožní vytvořit dynamické sekvence snímků. Editor by měl být zejména uživatelsky přívětivý, aby jej dokázal ovládat i člověk, který má pouze základní znalosti práce s počítačem, a současně by zkušenému uživateli měla tvorba obrazové sekvence trvat nejkratší možnou dobu s takovými výsledky, se kterými bude spokojen. V zadání je specifikován požadavek na efekty rotace pozorovatele (pan) a přiblížení (zoom). To je pro editor sekvencí nedostatečné zadání, proto je nutné dále rozšířit funkčnost o další efekty a možnosti.

Druhá kapitola se zabývá rozborem problematiky. Popisuje možnosti dnešních nástrojů na tvorbu sekvencí, dále popisuje funkci čoček, které se objevují v objektivech fotoaparátů. Podrobně rozebírá funkci optické soustavy a vlastnosti objektivů fotoaparátů. Také zde jsou uvedeny nejčastější vady obrazu, ke kterým dochází při průchodu obrazu optickou soustavou. Dále se krátce zmiňuje o křivkách pro výpočet dráhy kamery a také o transformaci souřadnic, která se využívá pro pohyb kamery.

Třetí kapitola je zaměřena na návrh implementace. Uvádí způsob transformace souřadnic fotografie do 3D prostoru, kterou provádí poskytnutá knihovna. Detailně popisuje návrh práce s kamerou a jejich požadovaných akcí pohybu (pan) a přiblížení (zoom). Obsahuje návrh struktury pro uložení veškerých informací použitých pro animaci a strukturu souboru pro uložení projektu. Dále vysvětluje výpočet dráhy kamery ze zadaných bodů pomocí křivek a možnost měnit rychlost pohybu kamery v závislosti na své pozici. Jsou zmíněny požadavky na implementaci uživatelského rozhraní a rozložení jeho ovládacích prvků. Popisuje také možnosti dalších rozšíření editoru, jako jsou přechodové a obrazové efekty. U těchto rozšíření navrhuje možné způsoby jejich implementace.

Čtvrtá kapitola se zabývá využitím knihovny pro transformaci souřadnic. Popisuje způsob vykreslování scény ze zadané fotografie a zadaných parametrů, dále způsob implementace kamery a jejího pohybu. Zmiňuje změny, které byly provedeny během implementace struktury pro uložení informací a také rozšíření návrhu u změn rychlosti pohybu, kdy byly zavedeny dva režimy práce. Uvádí provedené změny, které měly za cíl optimalizovat vykreslování. Je zde popsána implementace přechodových efektu a jejich možnosti. Také je zde informace o možnostech obrazových efektů a způsobu jejich vytvoření. Popisuje také výpočet jednotlivých kroků animace a nastavení scén při přehrávání. Ukazuje také výsledné uživatelské rozhraní a řešení komplikací vzniklých při jeho vytváření.

Pátá kapitola provádí otestování programu s ohledem na výkonovou náročnost výsledného produktu a požadavky na sestavu pro provoz programu.

Šestá kapitola v závěru zhodnocuje dosažené cíle, shrnuje výslednou práce a její možnosti.

V semestrálním projektu byl zpracován rozbor problematiky. Byly z něj převzaty kapitoly popisující obecně editory pro tvorbu sekvencí 2.1, čočky 2.2, objektivy fotoaparátu 2.3 a také geometrické transformace 2.5. Některé z těchto částí byly dále upraveny a doplněny. Dále byla v návrhu převzata část s transformací souřadnic 3.1 a pohybem kamery ve scéně 3.2. Dále byly využity kapitoly s návrhem uživatelského rozhraní 3.6 a dalších rozšíření 3.7. V implementaci byl v semestrálním projektu uveden pouze popis implementace transformace souřadnic a kamery. Většina uvedených částí byla dále upravena a rozšířena.

2 Rozbor problematiky

2.1 Editory pro tvorbu slideshow

Tvorba obrazových sekvencí (slideshow) je často využívána pro zobrazování prezentací. Může se jednat o prezentace firem, které zobrazují informace o své výrobě, různé spolky ukazující svou činnost, nebo pouze o rodiny, které se chtějí poděli se svými známými o zážitky z dovolené. Tyto prezentace je možné vytvořit nástrojem na tvorbu prezentací jako je Microsoft PowerPoint, případně Impress z kancelářského balíku OpenOffice. Tyto nástroje jsou zaměřeny na vytváření převážně textových prezentací doplněných obrazy a není primárně určen k zobrazování fotografií. Využití tohoto nástroje pro zobrazování obrazových sekvencí má omezené možnosti. Pro přidání další fotografie je nutné nejprve vytvořit novou stránku, na ni následně vložit požadovaný snímek a ten zvětšit na požadované rozměry. Pro vytváření rozsáhlejší prezentace fotografií jsou tyto nástroje nevhodné. Vytvoření dynamických efektů pro přiblížení a pohyb po snímku je v těchto programech obtížné. Existují také nástroje primárně určené k vytváření obrazových sekvencí. Mezi tyto patří i různé editory pro střih videa. Tyto nástroje mohou být k vytváření dynamických efektů uzpůsobeny. Často také obsahují další nástroje pro změnu barevného tónu, přidání audio souborů k prezentaci a také například jejich následný export do video-souboru. Tyto nástroje však zobrazují fotografie ve dvourozměrném systému tak jak jsou vyfoceny a uloženy v počítači. Avšak fotografie je vytvořená z reálného světa a ten je trojrozměrný. Použitím trojrozměrného editoru je možné částečně restaurovat původní stav reálného světa. Vytvořené sekvence budou tedy vypadat reálněji, zejména při pohybu ve scéně. U těchto editorů odpadá problém zobrazování panoramatických fotografií. Dvourozměrné editory jsou schopny zobrazit panoramatickou fotografii pouze jako 2D snímek. U 3D editorů je možné se po této panoramatické fotografii volně rozhlížet. Takové editory nejsou příliš rozšířené. Dále existují prohlížeče panoramatických fotek, které umožňují volné rozhlížení po fotografii, avšak neumožňují tvorbu animovaných sekvencí.

2.2 Tenká čočka

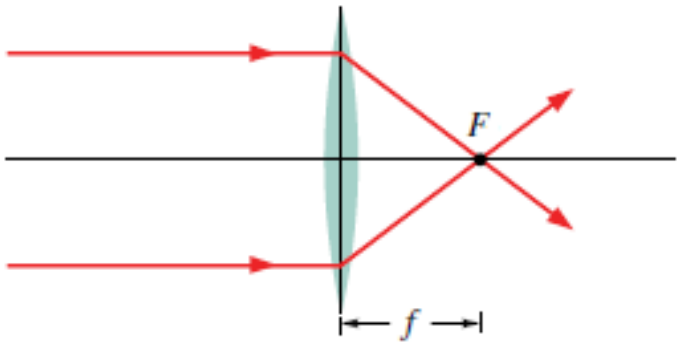
Pro přiblížení funkce fotoaparátu je nutné se nejdříve věnovat základům fyzikální optiky, ze kterých funkce objektivu fotoaparátu vychází.

Zdrojem pro popis tenkých čoček je [1], kde lze nalézt podrobnější informace. Definice čočky převzata z [1]: „Čočka je průhledné (transparentní) těleso se dvěma lámavými plochami, jejichž centrální osy splývají. Společná centrální osa je centrální osou čočky. Je-li čočka obklopena

vzduchem, láme se světlo ze vzduchu do čočky, prochází čočkou a znovu se láme do vzduchu. Při každém lomu se může změnit směr chodu světla.“

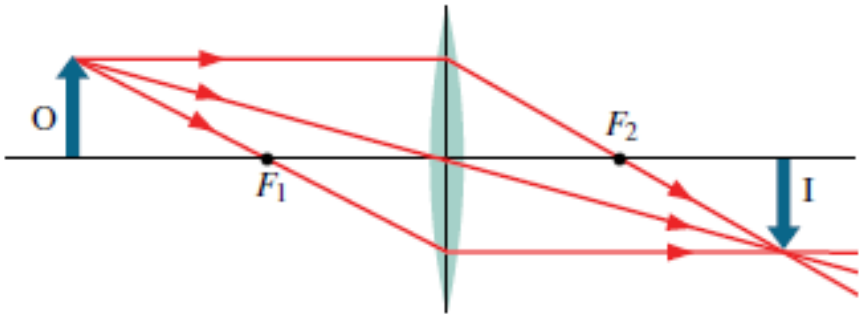
Pro zjednodušení se budeme věnovat pouze spojné čočce. Informace o rozptylné čočce lze nalézt v literatuře. U spojné čočky se rovnoběžné paprsky sbíhají v jediném bodě. Pro výklad předpokládáme tenkou čočku pro kterou platí, že nejtlustší část je tenká ve srovnání s předmětovou vzdáleností, obrazovou vzdáleností a poloměrem křivosti.

Procházejí-li spojnou čočkou rovnoběžné paprsky, sbíhají se v jediném bodě. Tento bod se nazývá ohnisko a je ve vzdálenosti *f* od středu čočky. Vzdálenost *f* se nazývá ohnisková vzdálenost. Ohnisko je znázorněno na obrázku 2.1, kde se dva rovnoběžné paprsky lámou směrem do ohniska *F* ve vzdálenosti *f*.



Obr. 2.1 Ohnisko spojné čočky, převzato z [1]

Při zobrazování čočkami může nastat několik možností v závislosti na pozici pozorovaného objektu. Objekt se může nacházet ve vzdálenosti větší než ohnisková vzdálenost, může se nacházet v ohnisku a nebo se může nacházet ve vzdálenosti menší než ohnisková vzdálenost. Pro potřeby popisu funkce objektivu fotoaparátu však předpokládáme objekt ve vzdálenosti větší než je ohnisková vzdálenost. Ohnisková vzdálenost má hodnotu několika desítek milimetrů a pokud pořizujeme fotografie, je scéna vždy dále než ohnisková vzdálenost. Proto zde budeme uvažovat pouze tuto variantu. Ostatní případy jsou popsány v literatuře. Objekt O je dále od středu čočky než bod F_1 a tedy ve vzdálenosti větší než ohnisková vzdálenost *f*. Čočka vytváří reálný převrácený obraz I na opačné straně čočky. Na obrázku 2.2 je znázorněna tvorba obrazu spojnou čočkou.



Obr. 2.2 Tvorba obrazu spojnou čočkou, převzato z [1]

2.3 Objektiv fotoaparátu

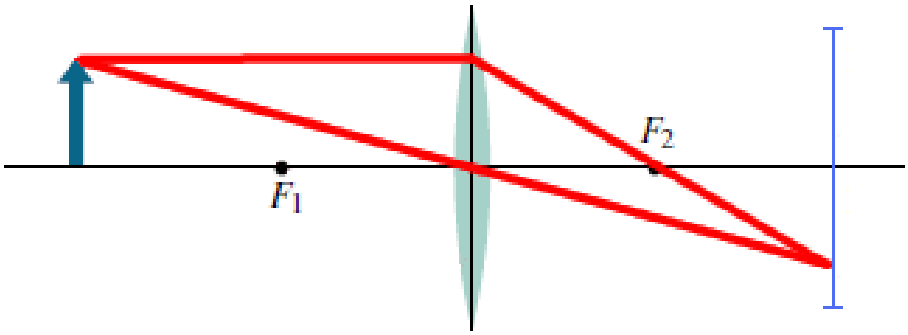
Zdrojem pro tuto kapitolu bylo [2]. Objektiv fotoaparátu je nejdůležitější částí fotoaparátu. Jedná se o soustavu čoček a zrcadel, které mají za cíl dosáhnou co nejmenší optické vady. A tím získat co nejkvalitnější fotografie. Pro objektiv je důležitá ohnisková vzdálenost, světelnost (clonové číslo) a zorný úhel. Světelnost objektivu není pro tuto práci důležitá, proto byl její popis vynechán.

2.3.1 Čočka

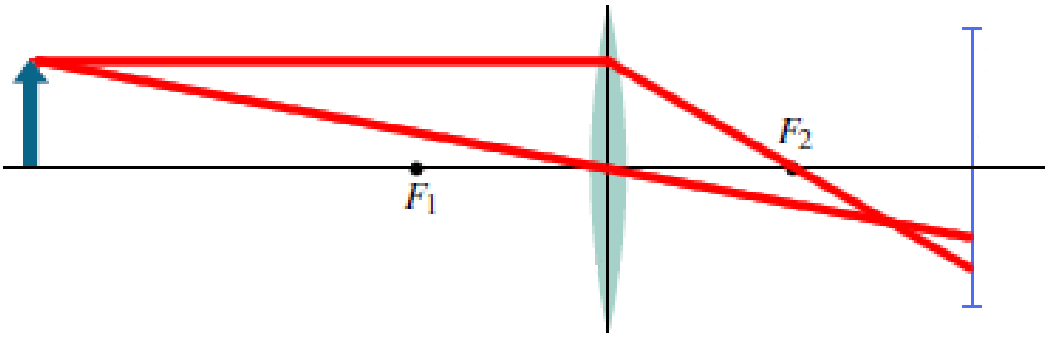
Fotoaparát obsahuje více čoček. Využívají se spojky i rozptylky. Menší počet čoček obsahují kompaktní přístroje které mají nižší rozměry na úkor výsledné kvality zobrazení. Na druhou stranu profesionální fotoaparáty obsahují velké množství čoček. Z fyzikálního hlediska jsou čočky podrobněji popsány v kapitole 2.2.

2.3.2 Zaostření obrazu

Při pořizování snímků je nutné fotoaparát zaostřit na fotografovaný objekt. Na obrázku 2.3 je zobrazený fotografovaný objekt a jeho obraz, který vzniká. Vzniklý obraz je přesně na pozici snímacího čipu a paprsky vyslané z objektu do různých směrů se opět sejdou ve stejném místě. Proto bude na výsledné fotografii objekt zaostřený. Na druhém obrázku 2.4 je fotoaparát ve stejné konfiguraci, ale fotografovaný objekt se posunul. Tím se změní pozice jeho obrazu mimo snímací čip a paprsky vyslané z různých směrů dopadají na snímacím čipu na různá místa. Proto bude obraz rozmazaný. Pro zaostření je nutné posunout optickou soustavu tak, aby se paprsky sbíhaly v jediném místě na snímacím čipu.



Obr. 2.3 správně zaostřený objekt

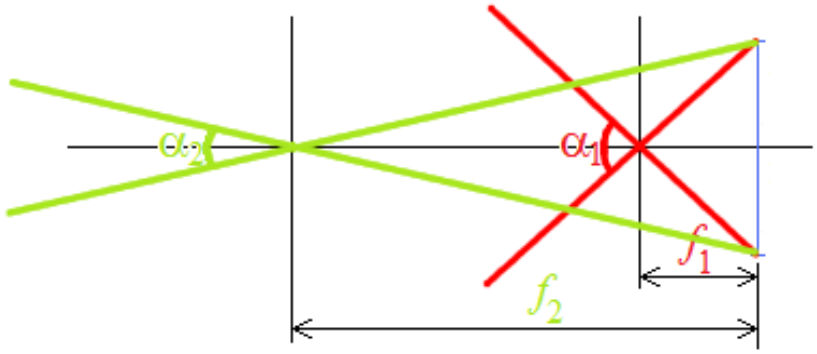


Obr. 2.4 nezaostřený objekt

2.3.3 Obrazový úhel

Obrazový úhel můžeme uvádět ve vodorovném či svislém směru a nebo ve směru diagonálním. V této části budeme předpokládat diagonální úhel. Je určen poměrem velikosti úhlopříčky obrazového čipu a ohniskové vzdálenosti objektivu. U klasického objektivu je obrazový úhel 40-60 stupňů. Změnou obrazového úhlu lze docílit u fotoaparátu funkce zoom, který se také označuje transfokátor. Při zvyšování obrazového úhlu dochází k oddalování a při snižování úhlu k přibližování. Při požadavku na změnu obrazového úhlu není možné měnit velikost snímacího čipu. Proto je nutné pro změnu obrazového úhlu měnit ohniskovou vzdálenost. U standardních fotoaparátů je možné měnit ohniskovou vzdálenost od 28mm až po 80mm.

Na obrázku 2.5 jsou zobrazeny dvě čočky každá s jinou ohniskovou vzdáleností. První čočka má ohniskovou vzdálenost f_1 a druhá ohniskovou vzdálenost f_2. Z obrázku je viditelný vliv ohniskové vzdálenosti na zobrazovací úhel. U menších ohniskových vzdáleností je úhel větší. Je nutné dodat, že toto znázornění platí pouze při zaostření do nekonečna, kdy platí, že vzdálenost čočky a snímacího prvku (obrazová vzdálenost) je rovna ohniskové vzdálenosti.



Obr. 2.5 Vliv obrazového úhlu na ohniskové vzdálenosti

Částečně lze měnit obrazový úhel pomocí snímacího čipu a to tím způsobem, že vybereme pouze určitou oblast na čipu. Tím se zmenší velikost čipu a tedy sníží obrazový úhel. Tento způsob je označován jako digitální zoom a není vlastností optiky fotoaparátu na rozdíl od optických transfokátorů. Nevýhodou tohoto řešení je že se zmenšuje rozlišení obrazu a tedy zhoršuje kvalita výsledného snímku. Proto není vhodné využívat digitální zoom pro pořizování kvalitních fotografii.

2.3.4 Širokoúhlý objektiv

Zdroj [3]. Tyto objektivy, jak už z názvu vyplývá dosahují zobrazení s velkým zorným úhlem. U širokoúhlých objektivů je stejný obrazový čip jako u klasického objektivu. Je tedy nutné zmenšit ohniskovou vzdálenost tak, aby úhel odpovídal poměru uhlopříčky a ohniskové vzdálenosti. Tyto objektivy mohou mít ohniskovou vzdálenost kolem 24mm, ale existují i širokoúhlé objektivy s ohniskovou vzdálenosti 21mm a méně. U širokoúhlých objektivů se však může projevovat nepřirozená perspektiva. Proto je vhodné používat tyto objektivy pouze za určitých podmínek.

2.3.5 Teleobjektivy

Zdroj [3]. Teleobjektivy mají menší obrazový úhel a tedy větší ohniskovou vzdálenost. To umožňuje fotografovat velmi vzdálené objekty. Teleobjektiv má ohniskovou vzdálenost kolem 135mm a výše. Teleobjektivy mají malou hloubku ostrosti a to umožňuje zobrazit ostré objekty s rozmazáním pozadí.

2.4 Optické vady objektivů

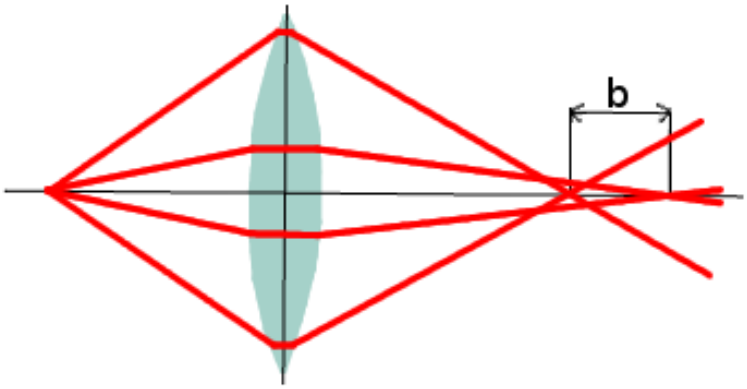
Optické vady jsou popsány v literatuře [4],[5]. U zobrazování objektů pomocí geometrické optiky s využitím čoček a zrcadel je předpoklad, že obraz reálného bodu je opět bod. Takto se chová ideální optický systém. V ideálním systému může být neostrost způsobena špatným zaostřením, případně pohybem předmětu či přístroje. U reálných systému však dochází k vadám objektivu. Pokud bychom byli schopni vytvořit ideální tenkou čočku, pak by se objektiv mohl skládat pouze z této jediné čočky.

Ideální čočku však není možné vyrobit. Proto se objektiv skládá ze systému čoček, který má za cíl se co nejvíce přiblížit ideální čočce, u které nedochází k optickým vadám. Ideální čočku nelze vytvořit ani pomocí soustavy čoček, protože některé vlastnosti jsou protichůdné a je nutné zvolit vhodný kompromis. Vady lze rozdělit na geometrické, chromatické a difrakční.

2.4.1 Otvorová vada (Sférická)

Paprsky vycházející z jednoho bodu na optické ose se nespojí v jediném bodu. Paprsky blíže optické osy se spojí ve větší vzdálenosti než paprsky které jsou odkloněny více. Velikost této chyby je daná délkou úsečky b obrázku 2.6, která udává rozdíl mezi maximální pozicí spojení paprsků u optické osy a minimální pozicí u paprsků s nejvyšším vychýlením.

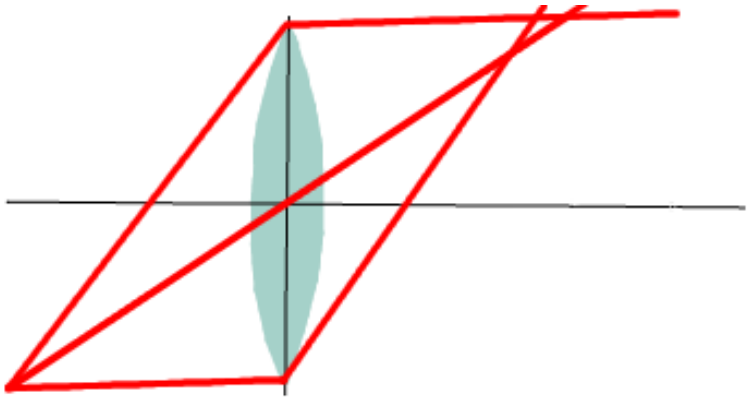
Tuto vadu lze eliminovat zacloněním, čímž se zmenší průřez paprsků. Zacloněním ale dojde ke ztrátě světelné energie. Dále lze vadu potlačit využitím rozptylky, která má opačný průběh. Je nutné tedy zvolit vhodnou kombinaci spojky a rozptylky.



Obr. 2.6 Otvorová vada

2.4.2 Asymetrická vada (Koma)

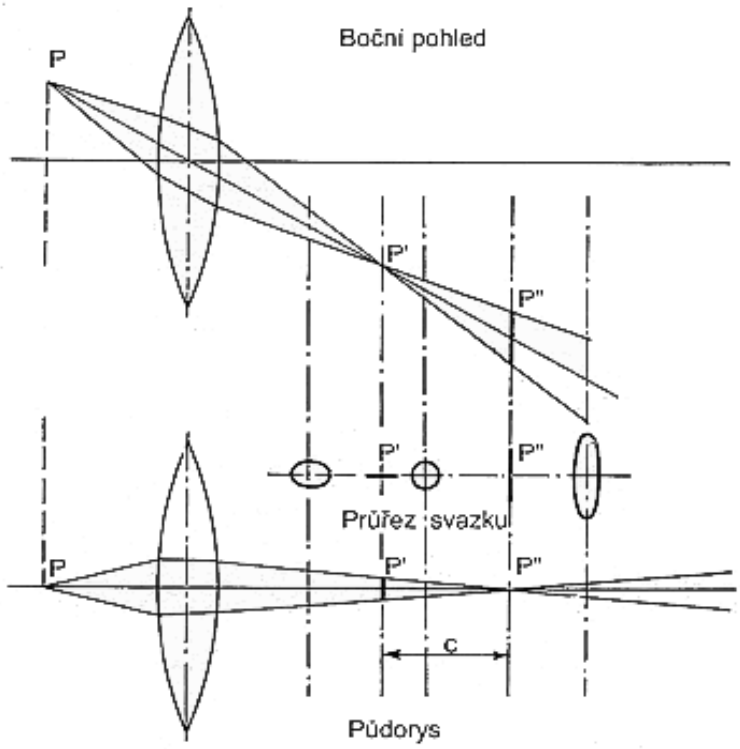
Koma stejně jako sférická vada vzniká při zobrazování širokým svazkem paprsků. V tomto případě však zobrazovaný bod leží mimo optickou osu. Obrázek 2.7. Body ležící mimo optickou osu se zobrazují jako skvrny, kdy skvrnou není kruhová ploška, ale útvar, který připomíná kometu. Jedná se o obálku množiny kružnic zvětšujících se poloměrů, jejichž středy leží na společné přímce. Proto se tato vada nazývá koma. Projevuje se již u bodů blízko optické osy. Proto je nutné tuto vadu spolu se sférickou vadou odstranit. Soustavy, které provádí korekci komy a sférické vady se nazývají aplanatické.



Obr. 2.7 Asymetrická vada

2.4.3 Astigmatismus

Tato vada se stejně jako koma projevuje při zobrazování bodů mimo optickou osu. Astigmatismus se však projevuje i u velmi úzkých svazků paprsků. Chová se tak, jako by měla čočka v různých řezech různé ohniskové vzdálenosti. Astigmatizmus je znázorněn na obrázku 2.8, kde u bočního řezu prochází paprsky bodem P' a u horního řezu prochází bodem P''. Ostatní paprsky vytvářejí svazek, jehož průřez má eliptický tvar. V bodech P' a P'' degeneruje elipsa na úsečku. Mezi body P' a P'' je místo, kde má průřez svazku kruhový tvar. Toto je místo nejmenšího rozptylu. Velikost chyby je určená vzdáleností bodů P' a P''. Z této vzdálenosti získáme astigmatický rozdíl. Vadu lze eliminovat použitím dvou soustav s opačným astigmatickým rozdílem.



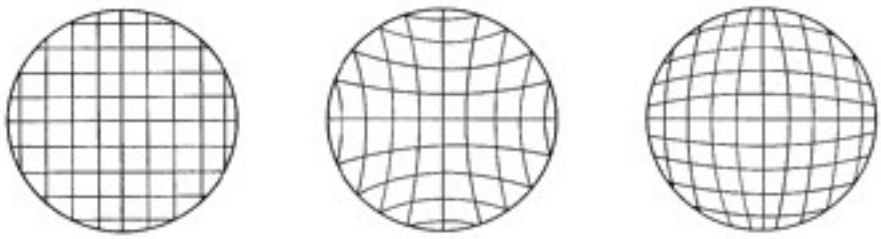
Obr. 2.8 Astigmatismus, převzato z [4]

2.4.4 Zklenutí obrazu

Projevuje se při zobrazování roviny kolmé na optickou osu pomocí kulové plochy. Snímání této roviny maticí kolmé k optické ose se nejostřejší obraz nevytvoří v rovině, ale na zakřivené ploše. Vzniká tak rozostřený obraz. Pokud je ostrý střed, pak jsou rozostřené okraje a pokud jsou ostré okraje, pak je rozostřený střed.

2.4.5 Zkreslení obrazu

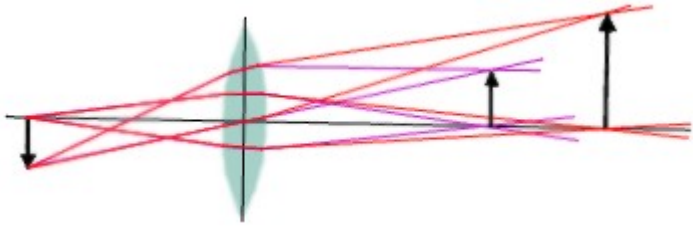
Zkreslení se projeví, pokud jsou různé body s různým příčným zvětšením jinak vzdálené od optické osy. Pokud zobrazíme kolmo na optickou osu pravoúhlou mřížku, může dojít k deformaci. Deformace je znázorněna na obrázku 2.9. Na prvním obrázku je nezkreslený obraz. V případě zkreslení obrazu jsou linie mřížky zobrazeny prohnuté. Na prostředním obrázku je zobrazeno poduškovité zkreslení, ke kterému dojde pokud příčné zvětšení roste se vzdáleností od optické osy. V případě, že příčné zvětšení klesá se zvyšující se vzdáleností od optické osy, pak dochází k soudkovitému zkreslení. Zkreslení se odstraní použitím opačné čočky, která zkreslení vykompenzuje. Čočku s poduškovitým zkreslením lze vykompenzovat použitím čočky se soudkovitým zkreslením a naopak.



Obr. 2.9 Zkreslení obrazu (nezkreslený, poduškovité, soudkovité), převzato z [4]

2.4.6 Barevná vada (Chromatická)

Při korekci geometrických vad jsme schopni tyto vady částečně korigovat, avšak korekce je pouze pro světlo o určité vlnové délce. Pro ostatní vlnové délky nemusí být korekce správná. Různé barvy se liší svou vlnovou délkou. Vlnová délka může být 400nm (fialová) až 700nm (červená). Současně platí, že index lomu na rozhraní je závislý na vlnové délce. Vyšší vlnová délka má menší index lomu a proto se méně láme. Ohnisková vzdálenost tedy není konstantní, ale mění se v závislosti na vlnové délce dopadajícího světla. Červený obraz je tedy zobrazen ve větší vzdálenosti než obraz fialový. Současně zvětšení obrazu závisí na ohniskové vzdálenosti a proto bude červený obraz větší než fialový. Znázornění je na obrázku 2.10. Korekce lze provést využitím dalších čoček, který rozdíl v indexech lomu eliminují. Vhodnějším řešením však může být využití zrcadlových objektivů. Chromatická vada se projevuje pouze při lomu světla. Zrcadlové objektivy využívají odrazu a ten není závislý na vlnové délce.



Obr. 2.10 Barevná vada

2.5 Geometrické transformace

Zdrojem pro tuto kapitolu je [6] a [7]. Geometrické transformace jsou jednou z nejčastěji využívaných operací v počítačové grafice. Cílem geometrické transformace je změnit pozici vrcholů v aktuálním souřadnicovém systému. Může se jednat o operace posunutí, rotace, zkosení, změna měřítka a další. Geometrické transformace je možné zapsat pomocí matice nebo výrazu. Obě možnosti jsou ekvivalentní a liší se pouze zápisem. U maticového vyjádření má matice posunutí jiný tvar než než matice rotace a další. To je nevýhodné, proto se zavádí homogenní souřadnice, které tento problém eliminují. Pro tuto práci je nejdůležitější rotace, proto zde bude popsána pouze tato operace. Další operace lze nalézt například v [6]. Pro popis rotace je nutné se nejprve věnovat homogenním souřadnicím.

2.5.1 Homogenní souřadnice

U homogenních souřadnic je k n-rozměrnému vektoru přidána váha w. Například pro 2D vektor [x, y] je rozšířen na vektor [x', y', w]. Platí, že x=x'/w a y=y'/w. Tímto vyjádřením získáme i další výhodu a to, že libovolný bod můžeme přesouvat po přímce protínající střed souřadného systému a zadaný bod. Při zvyšování váhy se bude bod posouvat blíže ke středu a při snižování váhy dále od středu. Tím lze bod umístit na libovolné místo na přímce zadané vektorem. Při limitě w jdoucí k 0 je výsledný bod v nekonečnu.

2.5.2 Rotace

Rotace je využívaná pro nastavení pozice kamery. Ve 2D prostoru stačí pro rotaci jedna matice. Jedná se o rotaci bodu [x, y] okolo středu souřadného systému o úhel α. Kladný směr otáčení je proti směru hodinových ručiček. Ve vzorci 2.1 je uvedena matice pro rotaci ve 2D. Pomocí této matice lze vypočítat novou pozici násobením matice rotace a bodem v homogenních souřadnicích. Výpočet je ve vzorci 2.2. Výpočet lze převést na ekvivalentní zápis ve vzorci 2.3.

$$R=\begin{bmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix} \tag{2.1}$$

$$[x',y',1]=[x,y,1]\cdot\begin{bmatrix}\cos\alpha & \sin\alpha & 0\\ -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix} \tag{2.2}$$

$$\begin{aligned}x' &= x\cdot\cos\alpha - y\cdot\sin\alpha\\ y' &= x\cdot\sin\alpha + y\cdot\cos\alpha\end{aligned} \tag{2.3}$$

Ve 3D si s touto jednou maticí nevystačíme. Je nutné vytvořit tři matice R_x, R_y, R_z pro rotace kolem jednotlivých os x, y, z v počátku souřadného systému.

$$R_x=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\alpha & \sin\alpha & 0\\ 0 & -\sin\alpha & \cos\alpha & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{2.4}$$

$$R_y=\begin{bmatrix}\cos\alpha & \sin\alpha & 0 & 0\\ 0 & 1 & 0 & 0\\ -\sin\alpha & \cos\alpha & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{2.5}$$

$$R_z=\begin{bmatrix}\cos\alpha & \sin\alpha & 0 & 0\\ -\sin\alpha & \cos\alpha & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{2.6}$$

2.6 Křivky

Křivky se využívají v počítačové grafice k mnoha účelům. Využívají se pro modelování, jsou s jejich pomocí vytvářeny fonty, nebo se používají k popisu dráhy pohybu objektů. V této práci jsou využity pro výpočet pohybu kamery. Zdrojem pro tuto kapitolu je [6]. Pro modelování se využívá polynomiálních křivek, které lze skládat na po částech polynomiální. Skládají se ze segmentů polynomiálních křivek u kterých zachováváme spojitost. Nejčastěji se používá spojitost třetího stupně tzv. kubiky. Křivky se skládají z několika řídících bodu, ze kterých je vypočítán průběh křivky. Využívají se dva typy křivek a to interpolační a aproximační. U interpolačních výsledná křivka prochází řídícími body. U aproximačních je křivka pouze ovlivněna řídícími body a nemusí jimi procházet. Křivek v počítačové grafice je velké množství, proto zde budou uvedeny pouze některé z křivek, které byly využity v této diplomové práci.

2.6.1 Catmul-Rom

Catmul-Rom křivky jsou popsány v literatuře [6] v kapitole Počítačová animace. Vychází z Kochanek-Bartels, proto je nejprve nutné popsat tento typ křivky. Křivka je zadána posloupností bodů P_0, P_1, …, P_N a dále koeficienty a, b, c pro každý z vnitřních bodů P_1, …, P_{N-1}. Křivka krajními body P_0 a P_N neprochází. Ty jsou využity pouze pro definici průběhu. Výpočet se provede pomocí výpočtu Hermitovských kubik podle vztahu 2.7. Pro výpočet jsou nutné dva tečné vektory v každém bodě. Pro segment nalevo a napravo od uzlu. Jsou to tzv. vstupní a výstupní vektory, které se vypočítají podle vztahu 2.8.

$$Q(t)=[t^3\,t^2\,t\,1]\begin{bmatrix}2 & -2 & 1 & 1\\ -3 & 3 & -2 & -1\\ 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}P_i\\ P_{i+1}\\ \vec{r}_i\\ \vec{l}_{i+1}\end{bmatrix} \tag{2.7}$$

$$\begin{aligned}\vec{l}_i&=\frac{(1-a)(1+b)(1-c)}{2}(P_i-P_{i-1})+\frac{(1-a)(1-b)(1+c)}{2}(P_{i+1}-P_i)\\ \vec{r}_i&=\frac{(1-a)(1+b)(1+c)}{2}(P_i-P_{i-1})+\frac{(1-a)(1-b)(1-c)}{2}(P_{i+1}-P_i)\end{aligned} \tag{2.8}$$

Pokud bude hodnota všech koeficientů a=b=c=0, pak je vstupní a výstupní vektor shodný. Hodnota vektorů pro body P_1, …, P_{N-1} je dána vztahem 2.9 a odpovídá tečnému vektoru křivky. Tento speciální případ Kochanek-Bartels křivky se nazývá Catmul-Rom. Upravením výpočtu 2.7 po dosazení tečných vektorů získáme vztah 2.10 pro výpočet křivky Catmull-Rom. V tomto případě je tvar křivky dán pouze řídícími body. Problémem u tohoto typu křivek může být to, že vypočítaná křivka nemusí ležet v konvexní obálce svých řídících bodů. Interpolaci počátečního a koncového bodu lze provést jejich opakováním.

$$\vec{l}_i=\vec{r}_i=\frac{P_{i+1}-P_{i-1}}{2} \tag{2.9}$$

$$Q(t)=[t^3\,t^2\,t\,1]\begin{bmatrix}-1 & 3 & -3 & 1\\ 2 & -5 & 4 & -1\\ -1 & 0 & 1 & 0\\ 0 & 2 & 0 & 0\end{bmatrix}\begin{bmatrix}P_{i-1}\\ P_i\\ P_{i+1}\\ P_{i+2}\end{bmatrix} \tag{2.10}$$

2.6.2 Beziérovy kubiky

Beziérovy křivky jsou popsány v literatuře [6] v kapitole Křivky a plochy. Beziérovy křivky jsou nejčastěji používané aproximační křivky. Beziérovy křivky jsou obecně n-tého stupně. Křivka n-tého stupně je určena n + 1 body. Využívá Bernsteinovy polynomy n-tého stupně B_i^n. Vztah 2.11. Beziérova křivka prochází prvním a posledním řídícím bodem. Další důležitou vlastností je, že výsledná křivka leží v konvexní obálce.

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-1}; i = 0,1,\ldots n \tag{2.11}$$

Nejčastějším případem jsou právě Beziérovy kubiky. Je to Beziérova křivka využívající Bernsteinovy polynomy stupně 3. Je tedy dána čtyřmi řídícími body. Výpočet lze provést dle maticového zápisu ve vztahu 2.12. Poté dosazováním parametru *t* získáme jednotlivé body křivky. Parametr *t* nabývá hodnot v intervalu <0,1>. K parametru *t* postupně přidáváme konstantní hodnotu *Δt* v závislosti na požadovaném počtu bodů křivky. Existuje také rekurzivní varianta tohoto algoritmu. V tomto případě však bude dostačující využití konstantní hodnoty *Δt*. Pro splnění spojitostí musí být koncový bod P_3 křivky prvního segmentu shodný s počátečním bodem Q_0 křivky druhého segmentu. Pro splnění shodných tečných vektorů musí bod $P_3=Q_0$ ležet na úsečce dané body P_2 a Q_1.

$$Q(t) = [t^3\, t^2\, t\, 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix} \tag{2.12}$$

3 Návrh implementace

3.1 Transformace 2D souřadnic do 3D prostoru

Důležitou součástí práce je převedení dvourozměrné fotografie do trojrozměrného prostoru. Při výpočtu je možné vycházet z principu objektivu. Objektiv byl popsán v kapitole 2.3. Pro zjištění 3D souřadnice je možné vypočítat paprsek vycházející ze zadaného bodu na čipu fotoaparátu a procházející středem čočky. Při fotografování se ztrácí informace o vzdálenosti bodu od čočky fotoaparátu. Proto již přesnou pozici bodu ve 3D souřadnicích nelze vypočítat. Řešením je namapování obrazu na jednotkovou kouli. Budeme předpokládat, že jsou všechny výsledné body ve vzdálenosti 1. Vzniklý obraz tedy nebude prostorový. Pokud předpokládáme, že se bude scéna snímat kamerou a následně zobrazovat na 2D zobrazovacím zařízení, pak není třeba získat prostorový obraz, protože projekcí by se tato informace opět ztratila. Omezením je to, že kamera musí zůstat na stejném místě a umožňuje pouze rozhlížení a přiblížení.

Popis transformace vychází z knihovny pro převod 2D souřadnic do prostoru. Knihovna byla poskytnuta vedoucím práce. Knihovna poskytuje také další typy objektivů. Popsána zde bude pouze ideální tenká čočka jako příklad řešení. Tato knihovna je také schopna opravit některé optické vady ke kterým dochází při promítání obrazu objektivem. Při požadavku na korekci optických vad konkrétního objektivu je nutné zjistit k jakým vadám v objektivu fotoaparátu dochází.

Při výpočtu předpokládáme ideální tenkou čočku. Cílem je najít místo průsečíku jednotkové koule a paprsku vyslaného ze zadaných souřadnic. Ze zadaných souřadnic x,y se vypočítá velikost diagonály d podle vztahu 3.1. Je nutné, aby maximální délka diagonály byla 1. Poté se z diagonály a zadané ohniskové vzdálenosti vypočítá diagonální úhel α podle vztahu 3.2. Diagonála se musí převést na skutečnou velikost diagonály pomocí násobení diagonálou kinofilmu. Uvažujeme-li standardní kinofilm, pak jsou jeho rozměry 36x24mm, pak je maximální velikost diagonály vypočítaná dosazením poloviny šířky kinofilmu a poloviny výšky kinofilmu do vztahu 3.2. Jedná se však pouze o polovinu skutečné diagonály, protože uvažujeme počátek souřadného systému uprostřed políčka kinofilmu. Výsledné souřadnice x,y se vypočítají podle vztahů 3.3 a 3.4. Nakonec je nutné vypočítat pozici na ose z. Při tomto výpočtu se využije funkce kosinus a hodnota souřadnice na ose z se vypočítá vztahem 3.5.

$$d = \sqrt{x^2 + y^2} \tag{3.1}$$

$$\alpha = \arctan \frac{d * 21{,}63}{f} \tag{3.2}$$

$$x' = \frac{\sin(\alpha) * x}{d} \tag{3.3}$$

$$y' = \frac{\sin(\alpha) * y}{d} \tag{3.4}$$

$$z' = \cos(\alpha) \tag{3.5}$$

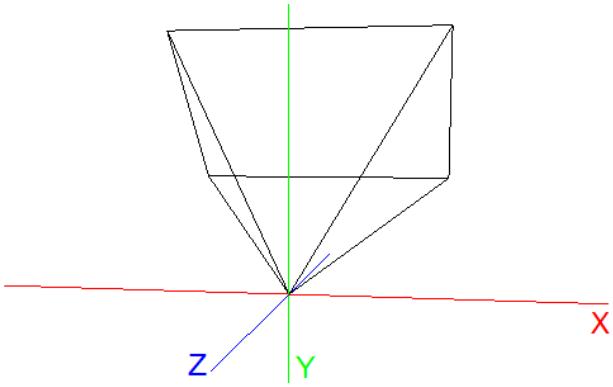
3.2 Kamera

Nastavení kamery je jednou z hlavních částí této práce. Je nutné, aby se chovala přirozeně a umožnila operace rotace a zoom. Obraz je staticky namapován na kouli o poloměru 1, která má střed shodný se středem souřadného systému. Tato koule zůstává stále na stejném místě a pohyb se provádí pomocí manipulace kamery. Kamera je umístěna ve středu koule a tedy také ve středu souřadného systému. Tím je umožněno libovolné rozhlížení do všech stran, zejména u panoramatických fotek. Díky namapování na kouli působí obraz reálněji než na původním dvourozměrném snímku. Podle zadání je nutné implementovat operace rotace a zoom. S kamerou není možné pohybovat do stran, protože tím by se kamera dostala mimo střed koule a došlo by ke zkreslení obrazu a ten by se stal nepoužitelným. Operací rotace a zoom pokryjeme tedy možnosti nastavení kamery ve scéně.

3.2.1 Rotace pozorovatele

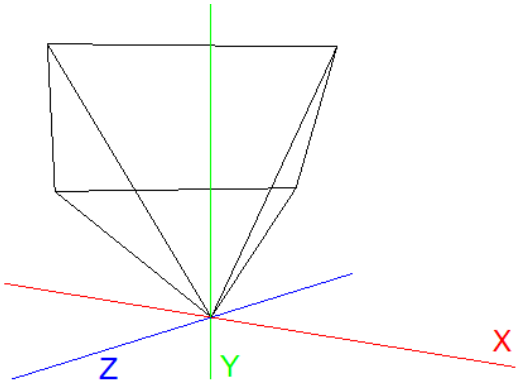
Rotace pozorovatele a tedy také kamery se provede pomocí popsaných geometrických transformací rotace. Pro pohyb kamery nahoru a dolů slouží rotace kolem osy x. To se provede podle dříve uvedené matice R_x ve vzorci 2.4 a pro pohyb vpravo a vlevo slouží rotace kolem osy y podle matice R_y ve vzorci 2.5. Tím dosáhneme natočení kamery na libovolnou pozici. Implementace bude provedena pomocí knihovny OpenGL a tato knihovna nemá funkce pro nastavení kamery. Tato kamera je umístěna fixně v souřadném systému. Manipulace s kamerou se tedy provádí pomocí manipulace objektů, kdy se nastavení kamery provede požadovaným natočením všech objektů scény. Nejedná se tedy o transformaci kamery, ale transformace bodů v souřadném systémy. Z důvodu lepší názornosti není popsána transformace kamery, ale transformace jehlanu, znázorňující kameru. Provádí se tedy rotace bodů jehlanu. Pořadí transformací je inverzní vzhledem k nastavení kamery.

Nejdříve se natočí kamera ve směru osy x, kdy se nastaví pohled ve svislém směru. Nastaví se tedy vychýlení kamery nahoru nebo dolů o zadanou hodnotu. Na obrázku 3.1 je obdélník, který znázorňuje pohled kamery.



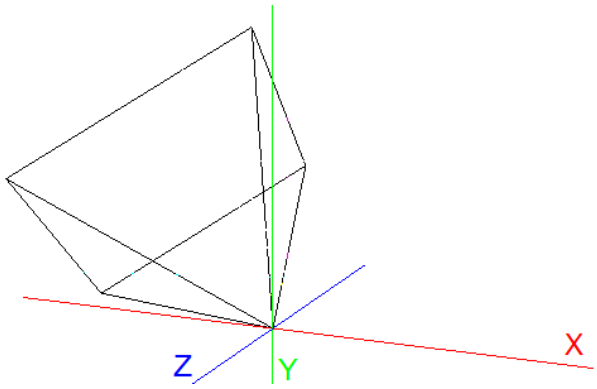
Obr 3.1 Natočení v ose x

Poté se kamera natočí kolem osy y a pohled se nastaví ve vodorovném směru. Kamera se otočí vpravo nebo vlevo. Obrázek 3.2



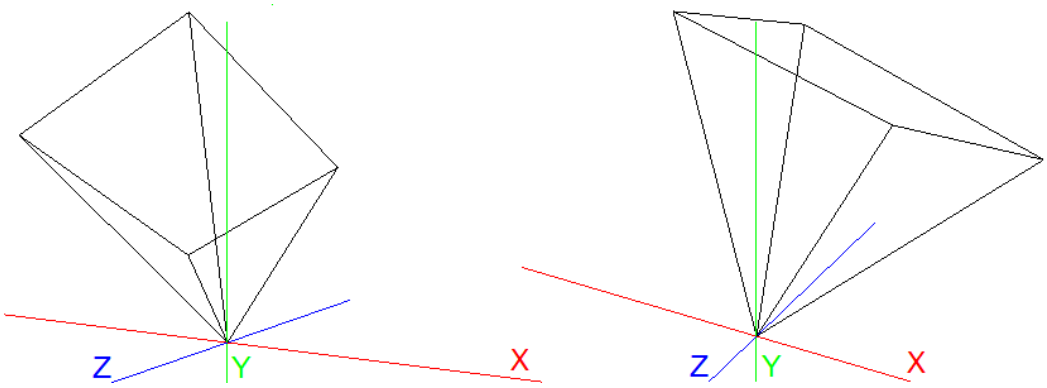
Obr 3.2 Natočení v ose y

Je nutné zachovat toto pořadí operací, protože pokud by se nejprve rotovala kamera kolem osy y a pak teprve kolem osy x, tak by došlo s rotací k naklonění kamery. Čím větší by byla rotace kolem osy x, tím by se zvětšovalo také naklonění. Nejdříve je na obrázku 3.3 provedena rotace kolem osy y a po rotaci kolem osy x je na znázorněném pohledu pohledu jasně viditelný jev naklánění při chybném pořadí transformací. Proto je nutné pořadí transformací zachovat.



Obr. 3.3 Chybné pořadí rotací

Rotace může být dále rozšířena o rotaci kolem osy z. Matice rotace R_z je uvedena ve vzorci 2.6, ta může sloužit k záměrnému naklánění kamery do stran. Tato operace se provádí jako první. V případě, že by se prováděla později, došlo by chybnému nastavení kamery.



Obr 3.4 Srovnání správné a chybné rotace.

Na obrázku 3.4 vlevo bylo naklonění provedeno jako první a výsledný obraz je správný. Na obrázku 3.4 vpravo je chybné pořadí transformací, kdy byla provedena rotace kolem osy z jako poslední a došlo k chybnému nastavení.

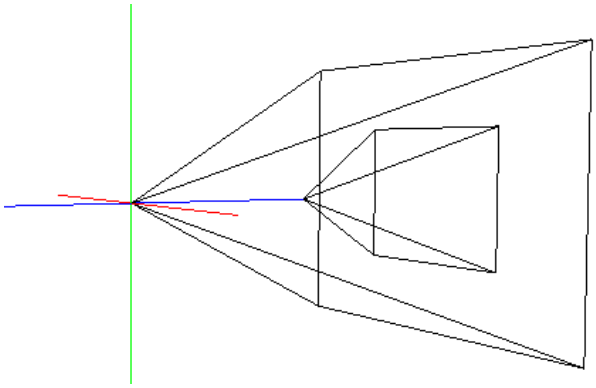
Pomocí těchto transformací lze nastavit libovolné natočení kamery ve všech směrech. Důležité pro implementaci je zachovat požadované pořadí provádění jednotlivých geometrických transformací. V případě nedodržení pořadí by docházelo k chybnému nastavení kamery. Pořadí transformací by tedy mělo být natočení kolem osy z, následně kolem osy x a nakonec kolem osy y. Pro nastavení skutečné kamery tedy platí pořadí inverzní. Nejdříve se kamera natočí do požadovaného směru kolem osy y, poté se nastaví pozice kamery nakloněním nahoru či dolů rotací kolem osy x a nakonec by se otočila kolem své osy tedy kolem osy z. Pro kompletní funkce kamery je nutné ještě popsat realizaci operace zoom.

3.2.2 Zoom

Pro implementaci operace zoom je možné využít více způsobů. Zde budou popsány dva způsoby realizace této operace. První z nich je posunutí kamery a druhý způsob je změna zorného úhlu kamery.

3.2.2.1 Zoom posunutím kamery

Kamera je definována svou pozicí a vektorem udávajícím směr pohledu. Koule, na kterou je promítán obraz má poloměr 1 a pokud je vektor normalizován, má také jednotkovou délku. Lze s výhodou využít váhy v homogenních souřadnicích. Vektor pohledu je vydělen váhou a tento výsledek je uložen jako pozice umístění kamery. Kamera se při změně váhy pohybuje po přímce dané vektorem pohledu. Váha se mění v intervalu $<1,\infty)$. Pokud nebudeme pracovat s váhou w, ale s hodnotou inverzní $w^{-1}=1/w$, nebude se vektor pohledu dělit, ale násobit. Pak bude tedy w^{-1} v intervalu $(0,1>$. U hodnot jdoucích k nule se přesouvá kamera směrem ke středu souřadného systému a dochází tedy k oddalování pohledu. U hodnot, blížících se k hodnotě jedna se kamera přibližuje ke kouli s obrazem a dochází tedy k přibližování obrazu. Při hodnotě $w^{-1}=1$ se však kamera již dostává na obvod koule a tedy mimo nanesený obraz. Aby bylo možné se dostat do počátku souřadného systému je nutné změnit omezení intervalu na $<0,1)$.

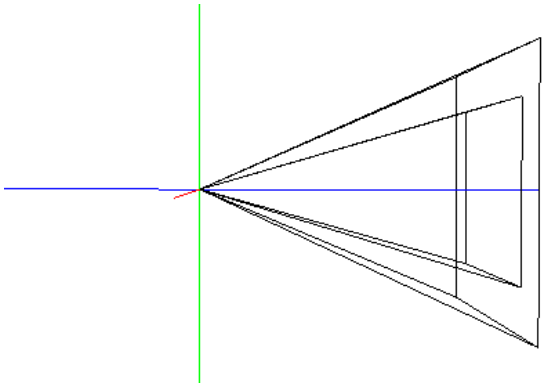


Obr. 3.5 Posunutí kamery

Obrázek 3.5 znázorňuje přiblížení při různém nastavení kamery. Na obrázku jsou dvě kamery se stejným obrazovým úhlem, kdy díky posunutí kamera zobrazuje menší oblast a obraz je tedy přiblížen. Výhodou tohoto řešení je jednoduchost implementace. Nevýhodou je, že toto řešení není úplně správné, kdy kamera by měla být umístěna ve středu souřadného systému. Obrazový výstup tohoto řešení by však měl být podobný jako u následující metody. Omezené je zde pouze maximální přiblížení, které závisí na nejvzdálenější možné pozici kamery, která je v tomto případě omezena. Pro reálné využití však tohoto omezení nebude dosaženo a přiblížení bude dostatečné. I při omezení je stále možné přiblížit se na úroveň jediného případně velmi malého počtu pixelů zdrojového obrazu. Snímky nebudou v takovém rozlišení, aby toto omezení mohlo působit problémy při zobrazení.

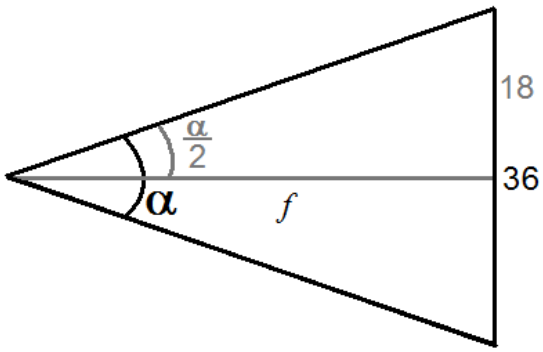
3.2.2.2 Zoom změnou zorného úhlu

Druhou možností, jak implementovat zoom kamery je změnou zorného úhlu kamery. Při zvyšování zorného úhlu dochází k oddalování obrazu, při snižování zorného úhlu k přibližování.



Obr. 3.6 Změna úhlu kamery

Na obrázku 3.6 jsou znázorněny různé zorné úhly a jejich vliv na přiblížení obrazu. U tohoto způsobu je možné zorný úhel vypočítat pomocí ohniskové vzdálenosti. Výpočet se provede dosazením ohniskové vzdálenosti do trojúhelníku. Známe ohniskovou vzdálenost a rozměr kinofilmu. Jak bylo dříve uvedeno, u fotoaparátů se neudává reálná ohnisková vzdálenost, ale ohnisková vzdálenost vztažená k rozměrům kinofilmu. Proto můžeme předpokládat rozměry kinofilmu jako konstantu.



Obr. 3.7 Schéma výpočtu obrazového úhlu

Z obrázku 3.7 je patrné, že zorný úhel α lze vypočítat z ohniskové vzdálenosti f a rozměrů políčka kinofilmu. V závislosti na nastavení úhlu kamery se bude lišit zadaná konstanta. Pokud se pro nastavení kamery bude zadávat úhel kamery ve vodorovném směru, do vzorce se dosadí šířka políčka kinofilmu 36mm. Úhel kamery se potom vypočítá pomocí vztahu 3.6, ze kterého po úpravě dostaneme vzorec 3.7.

$$\tan\frac{\alpha}{2}=\frac{\frac{36}{2}}{f} \tag{3.6}$$

$$\alpha=2\cdot\arctan\frac{18}{f} \tag{3.7}$$

Tento způsob je blíže reálnému způsobu přibližování ve fotoaparátech. Přesto, že je tento způsob mírně složitější než předchozí metoda, je stále velmi jednoduchý. Výhodou tohoto řešení je, že umožňuje téměř nekonečně velké přiblížení, které závisí pouze na možnosti reprezentovat velmi malé hodnoty úhlu v počítači a na možnostech nastavení kamery v knihovně OpenGL. Další výhodou je možnost hodnotu přiblížení ukládat pomocí ohniskové vzdálenosti, která je pro uživatele výhodnější a více vypovídající než hodnoty přiblížení u předchozího způsobu.

3.3 Práce s informacemi

Je nutné zvolit vhodnou strukturu pro uložení všech informací o animaci. Všechny informace o slideshow budou uloženy v této struktuře. Při práci s uživatelským rozhraním budou jednotlivé komponenty přímo spolupracovat se zvolenou strukturou. Budou voláním funkci přidávat a mazat prvky a také modifikovat informace v nich obsažené. Při modifikaci některých informací je nutné aktualizovat i jiné komponenty uživatelského rozhraní. To by mohlo vést k chybám, kdy jedna komponenta aktualizovala komponentu jinou, ale při uložení informace by došlo k neplatné operaci, kterou struktura nedovoluje a vznikla by nekonzistence. Informace zobrazené uživateli prostřednictvím rozhraní by neodpovídaly uložené skutečnosti. Všechny změny by tedy musely být testovány, zda mají správné parametry. Komponenty není nutné aktualizovat přímo, ale je možné zavolat pouze funkci pro aktualizaci a tato komponeta si informace načte z právě aktualizované struktury. Tím lze zajistit, že všechny informace zobrazované uživatelským rozhraním odpovídají informacím uloženým. Při přehrávání animací budou všechny informace načítány z této struktury čímž bude zajištěno, že všechny změny provedené v uživatelském rozhraní jsou provedeny a přehrávaná animace je aktuální. Je nutné ji vhodně zvolit, aby byla přehledná a umožňovala jednoduchou změnu všech nastavení a současně by měla obsahovat všechny informace důležité pro přehrávání a editaci. Současně by měla zůstat konzistentní, aby nebylo možné dostat obsažené informace do nežádoucího stavu zadáním nepovolených parametrů. Také je nutné aby byla maximálně nezávislá na konkrétním uživatelském rozhraní a zvolené platformě. Následující zvolená struktura by měla fungovat v různých prostředích bez nutnosti větších změn. Navrženou strukturu lze rozdělit na 3 hlavní částí a to na globální informace o slideshow, informace o jednotlivých scénách a informace o pozicích kamery.

3.3.1 Globální informace

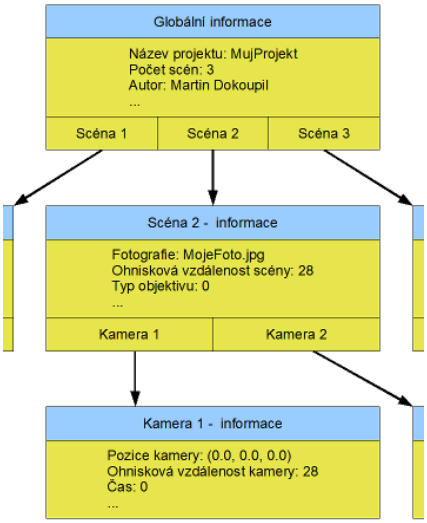
V této části jsou uloženy informace společné pro celou slideshow. Jedná se o hlavní část, ze které je možné přistupovat k částem následujícím. Mohou se zde nacházet například údaje o uživateli jako jméno autora, název projektu, dále informace o kompletní cestě k použitým snímkům a další informace. Nejdůležitější částí je seznam scén, kdy každý záznam seznamu obsahuje informace o každé ze scén.

3.3.2 Informace o scénách

Scénou v tomto případě budeme uvažovat fotografii uloženou ve trojrozměrném prostoru. Jsou zde obsaženy informace týkající se konkrétní scény. Je zde uložen název fotografie, ze které je scéna vytvořena. Protože je nutné s fotografií pracovat častěji, musela by se při každém požadavku načítat znovu. Proto je fotografie na počátku načtena a při dalším požadavku na fotografii je již připravena a není nutné čekat na načtení. Tím lze výrazně urychlit práci s fotografií. Jsou zde také uložené informace o pořízení fotografie a to ohnisková vzdálenost a typ objektivu, kterým byla fotografie vytvořena. Dále obsahuje seznam, kde jsou uloženy všechny pozice kamery dané scény. Při inicializaci je vytvořen jeden záznam o pozici kamery a to v čase 0, který udává počáteční nastavení kamery. Tento záznam není možné smazat, je možné pouze změnit jeho počáteční pozici.

3.3.3 Informace o pozici kamery

Zde je uložena pozice kamery, která udává úhel natočení kolem jednotlivých os x,y,z a hodnotu ohniskové vzdálenosti virtuální kamery. Dále zde je hodnota času, ve kterém se kamera na zadané pozici nachází. S těmito hodnotami pracuje engine pro přehrávání animací, který umísťuje kameru tak aby odpovídala požadavkům uvedeným v těchto záznamech.



Obr. 3.8 Struktura pro uložení

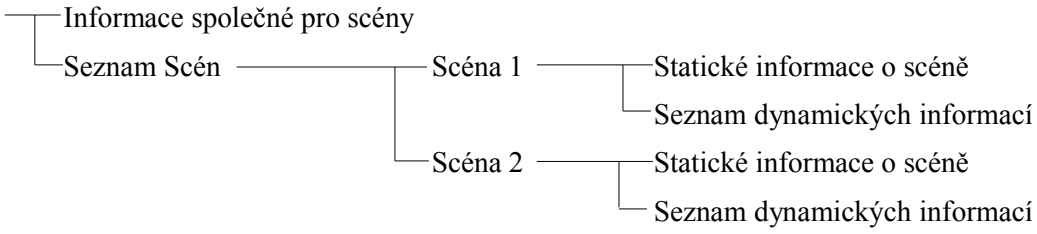
Výsledná navržená struktura je na obrázku 3.8. Při práci se strukturou se přistupuje s pomocí metod. Při požadavku na změnu parametrů scény je nejprve scéna získána ze seznamu. Každá scéna je označena identifikátorem, kde identifikátor udává index do seznamu všech scén. Takto získanou scénu je možné dále modifikovat pomocí metod. Stejným způsobem lze také pracovat s jednotlivými pozicemi kamery, kdy u vybrané scény se identifikátorem odkážeme na požadovanou pozici v seznamu a s tou následně pracujeme.

Informace ve struktuře lze jednoduše dále přidávat. Při přidání dalších možností animace se přidají další ukládané informace a metody pro čtení a modifikaci dané informace. Informaci je nutné uložit do té části, které se nově přidaná funkce týká. Správné umístění se určí podle toho, zda je přidaná funkce vlastností celé animace, konkrétní scény nebo pouze jedné z částí scény.

3.3.4 Uložení sekvencí

Po dokončení tvorby obrazové sekvence je nutné výsledek uložit, aby bylo možné se sekvencí dále pracovat. Je nutné zvolit vhodný konfigurační soubor. Do tohoto souboru lze následně uložit všechny informace o použitých snímcích a jejich vlastnostech (jako například ohnisková vzdálenost fotoaparátu při pořízení, poměr stran fotografie). Dále je nutné ukládat informace o pohybu kamery ve scéně, informace o době trvání animace, název a autor prezentace a další informace. Nejprve je nutné navrhnout strukturu tak aby byla přehledná, jednoznačná a měla by umožňovat jednoduché přidávání dalších rozšíření do struktury.

Celá animace se tedy dá rozdělit na jasně odlišitelné části. Lze ji rozdělit na informace společné pro všechny scény a seznam scén. Každá scéna lze dále rozdělit na statické informace o nastavení scény a dále seznam informací, které se ve scéně mění. V informacích pro všechny scény to mohou být například autor, název projektu, celková délka prezentace. Mezi informacemi o nastavení scény je název zdrojové fotografie nebo doba setrvání ve scéně. Dynamickými informacemi muže být pohyb kamery ve scéně v závislosti na čase. Možná struktura uložení je na obrázku 3.9 a odpovídá způsobu uložení informací v počítači.



Obr. 3.9 Schéma uložení do souboru

Pokud bude zachována zobrazená struktura, bude možné jednoduše přidávat další možnosti. Při rozšíření funkcionality se pouze přidá další položka do informací, ke kterým se vztahuje. Tato

struktura lze vhodně reprezentovat pomocí xml souboru, kdy jednotlivé části budou uzly xml stromu. Pro využití xml stromu není nutné programovat vlastní nástroje, protože pro práci s xml existuje velké množství nástrojů, které jsou volně k dispozici. Tyto nástroje umožňují načítat požadované uzly ze zadaného xml stromu. Příklad struktury převedené do xml formátu se nachází v příloze A. Jedná se o jeden z možných konfiguračních souborů pohybu kamery.

Hlavička head je pro společné informace. Dále následují informace o scénách, kdy každá scéna má svůj identifikátor. Každá scéna má svou hlavičku shead, kde jsou statické informace o dané scéně a následuje seznam dynamických informací. V tomto případě pohyb kamery move. Pokud má vlastnost pouze jeden atribut, je možné jej uložit jako obsah uzlu xml stromu, pokud má atributů více, uloží se jako atributy uzlu xml stromu takže není omezen počet atributů které je nutné předat. Jak již bylo zmíněno, při rozšíření funkcionality se pouze přidá další uzel do xml stromu. Další výhodou tohoto způsobu může být, že přehrávač hledá pouze ty uzly, pro které má definované chování. Pokud by vznikla nová verze editoru, která přidává další funkce, může starší verze stále pracovat se soubory vytvořenými ve verzi nové. Stará verze pouze přeskočí nové funkce, které se zde neuplatní. Tato výhoda se v případě této diplomové práce nevyužije, ale v případě reálného nasazení může mít tato dopředná kompatibilita význam. Současně platí také zpětná kompatibilita, kdy soubory ze starší verze mohou být spuštěny ve verzi nové. Nové části, které nejsou v konfiguračním souboru uvedeny se nastaví do implicitní hodnoty.

3.4 Dráha kamery

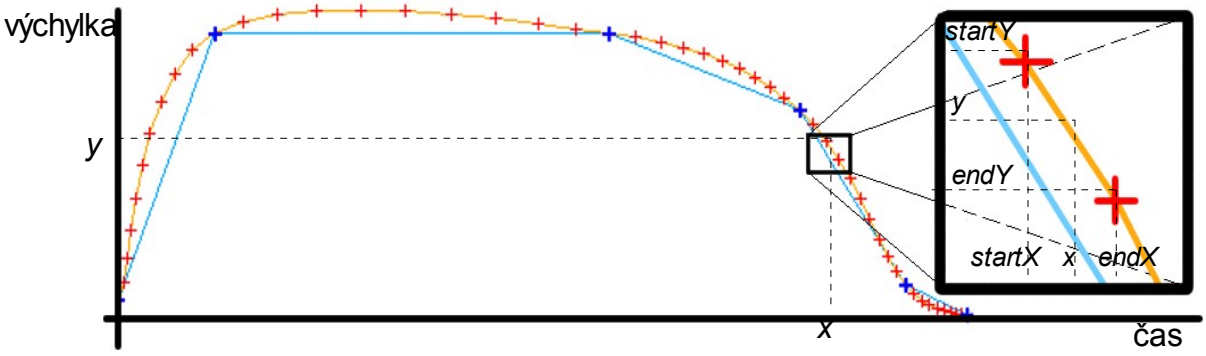
Pro výpočet dráhy kamery během animací je využito křivek. Dráha je vypočítána ze zadaných pozic kamery v definovaných okamžicích. Každá zadaná pozice je definována třemi úhly natočení kolem jednotlivých os, hodnotou ohniskové vzdálenosti a časem ve kterém je pozice nastavena. Při výpočtu dráhy využijeme dvou skutečností a to, že potřebujeme vždy získat konkrétní hodnotu otočení a přiblížení a také že úhly natočení a přiblížení jsou na sobě nezávislé. Proto se vytvoří čtyři křivky, které budou znázorňovat průběh otočení kolem osy x v závislosti na čase, kolem osy y a z v závislosti na čase a také přiblížení (ohniskovou vzdálenost) v závislosti na čase. Každá z veličin bude mít tedy definován svůj graf průběhu. Poté pro nastavení kamery v konkrétním čase se nalezne čas na ose x grafu vypočítaných bodů a žádaná vlastnost se vyčte na ose y. Takto se načtení provede pro všechny čtyři hodnoty, ze kterých se nastaví pozice kamery. Pokud není zadaná hodnota času na ose x vypočítaných bodů grafu nalezne se nejbližší levý a pravý soused a výsledná hodnota se získá lineární aproximací ze vztahu 3.8, kde *startY* je hodnota výchylky levého sousedního bodu křivky, *endY* je hodnota výchylky pravého sousedního bodu. *StartX* je čas výskytu levého souseda, *endX* je čas výskytu pravého souseda, *x* je aktuální čas a *y* je výsledná výchylka kamery.

$$y = startY + (endY - startY)\frac{x - startX}{endX - startX} \tag{3.8}$$

Tento způsob také umožňuje snadno změnit typ křivky. Řídící body kamery jsou zkopírovány jako řídící body křivky. Současně také výstup po výpočtu křivky je seznamem řídících bodů kamery. Jediným rozdílem je funkce která provádí výpočet bodů křivky. Lze tedy vytvořit libovolný počet metod pro výpočet křivek ze zadaných bodů. Poté na základě požadovaného typu křivky se zavolá příslušná metoda a následující práce již na použité metodě nezávisí. V případě že není zájem o využití křivek, ale pouze o provedení lineární interpolace jednotlivých zadaných bodů kamery, není nutné měnit výpočet. Pouze se nezavolá metoda pro výpočet křivky a jednotlivé řídící body kamery odpovídají bodům grafu. Proto je možné pracovat stejným způsobem jak s využitím křivek, tak bez nich.

Předpoklad je, že nejčastěji bude využita lineární interpolace bodů a interpolační křivka, protože kamera bude přesně procházet body zadanými uživateli. Jako interpolační křivka byl zvolen Catmul-Rom, který vychází z Kochanek-Bartels. Na rozdíl od něj je však jednodušší a tím že má parametry zadané automaticky, nemusí je zadávat uživatel. Předpoklad je, že uživatel nemá přehled o křivkách v počítačové grafice. Dalším důvodem je, že není nutné nastavovat naprosto přesně dráhu kamery. Pokud potřebuje uživatel zpřesnit dráhu může to provést přidáním dalšího řídícího bodu. Zadávání parametrů by mohlo vést ke složitějšímu ovládání a zmatenosti uživatele. Nevýhodou implementované interpolační křivky je, že výsledný průběh není v konvexní obálce řídících bodů. Pokud by uživateli tato skutečnost vadila, je využito také aproximační křivky, u které je výsledná křivka uvnitř konvexní obálky, ale v tomto případě nebude kamera přesně procházet zadanými body.

Na obrázku 3.10 je zobrazena interpolační křivka vypočítaná z řídících bodů a jsou zde vyznačeny proměnné využívané při výpočtu pozice.

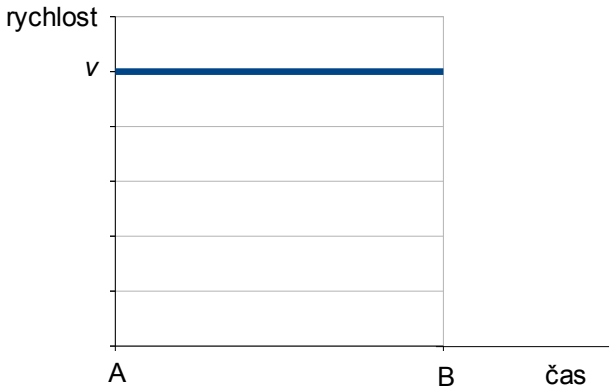


Obr. 3.10 Catmull-Rom ze zadaných bodů

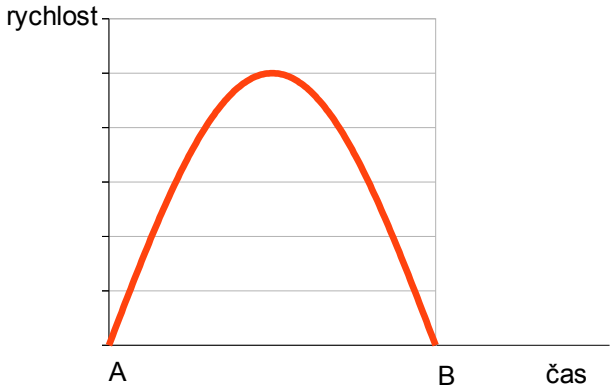
3.5 Průběh rychlosti pohybu

Během animace dochází k pohybu kamery po vytyčené dráze dané počátečním a koncovým bodem. Rychlost je závislá na zadané době přesunu z pozice A do pozice B a jejich vzdálenosti. Rychlost lze tedy vypočítat jako poměr vzdálenosti d a doby t dle vzorce 3.9. Kamera by se tedy pohybovala v každém úseku vypočítanou konstantní rychlostí, obrázek 3.11. Během přehrávání je vhodné mít možnost měnit průběh rychlosti kdy se během pohybu z bodu A do bodu B rychlost kamery mění. Jednou z funkcí, které jsou vhodné pro využití je průběh funkce sinus, kdy její charakteristika může odpovídat jednomu z možných průběhu kamery. Na počátku se kamera pohybuje pomalu a postupně zrychluje. Po dosažení poloviny dráhy k bodu B začíná zpomalovat, až pomalu dokončí pohyb v bodě B. Tento pohyb může působit v některých případech přirozeněji než pohyb konstantní rychlostí, kdy se kamera po dojetí do bodu B okamžitě zastaví. Což neodpovídá realitě, protože rozpohybovaná kamera má setrvačnou sílu a okamžité zastavení je obtížně proveditelné. Toto je pouze jeden z příkladů, jakou funkci implementovat pro průběh rychlosti kamery a implementace by měla umožňovat jednoduché přidávání dalších funkcí, které pohyb kamery popisují.

$$v = \frac{d}{t} \tag{3.9}$$



Obr. 3.11 Graf konstantní funkce



Obr. 3.12 Graf funkce sinus

Předpokládejme tedy funkci sinus jako průběh rychlosti pohybu. Využijeme pouze jednu polovinu této funkce. Průběh je znázorněn na grafu 3.12. Na ose x je čas, ve kterém se právě pohyb nachází a na ose y je rychlost pohybu kamery. Čas je relativní k okamžiku, kdy je kamera v bodě A. Je v rozsahu <0,max>, kde max je doba za kterou se má kamera dostat do bodu B. Musíme vzít v úvahu, že tato doba závisí na zadání od uživatelů a není konstantní. Proto je nutné čas normalizovat do pevně definovaného rozsahu. Nejvhodnější je zvolit rozsah <0,1>. Převod do normalizovaného rozsahu se provede vydělením aktuálního času hodnotou max a z normalizovaného do původního vynásobením hodnotou max. To umožňuje snadný převod těchto rozsahů. Rychlost pohybu na ose y není možné jednoduše převést do vhodnějšího rozsahu. Proměnná rychlost na ose y musí splnit podmínku, že v čase max se kamera nachází v bodě B. Rychlost je závislá na době nutné k přechodu, vzdálenosti mezi body a konkrétním tvaru průběhu. Proto nejsou na ose y grafu 3.11 a 3.12 konkrétní hodnoty. Důležitý je v tomto případě pouze průběh a ne konkrétní hodnoty.

Musíme upravit výpočet tak, aby byl nezávislý. Pokud přihlédneme k předchozí části 3.4, kdy dráha kamery je popsána v závislosti pozice na čase. Není možné vypočítávat křivky závislostí pro každé z použitých průběhů rychlosti. Jednodušší je přepočítání skutečného času na čas odpovídající požadované rychlosti pohybu. Normalizovaný čas v intervalu <0,1> je převeden na jiný čas ve intervalu <0,1> a tím lze docílit různé rychlosti pohybu v různých částech. Funkce převodu je závislá na požadovaném průběhu rychlosti pohybu.

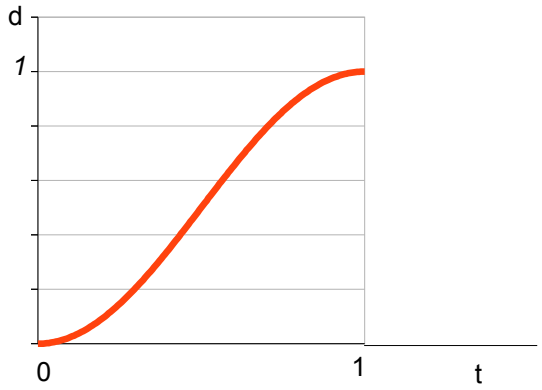
Máme tedy graf závislosti rychlosti na čase. Využijeme toho, že rychlost je derivací dráhy a tedy dle vztahu 3.10 se dráha vypočítá jako integrál rychlosti. V případě uvedené funkce sinus se vypočítá graf dráhy v závislosti na čase dle vztahu 3.11. Stejně jako byl čas normalizován na rozsah <0,1>, tak se stejným způsobem převede i dráha do rozsahu <0,1>. Po upravení normalizovaného vstupu na vstup pro funkci cosinus a po upravení výstupu funkce cosinus na normalizovaný tvar dostaneme výsledný vztah 3.12. Funkce je znázorněna na grafu 3.13. Tato funkce převede zadaný čas podle požadované transformace a kamera se tedy pohybuje podle zadané funkce. Každá funkce je

nezávislá na zadané vzdálenosti mezi body a době nutné k přechodu. Rychlost odpovídá zadané charakteristice a zachovává požadované podmínky na počátek a konec pohybu kamery.

$$d = \int v\,dt \tag{3.10}$$

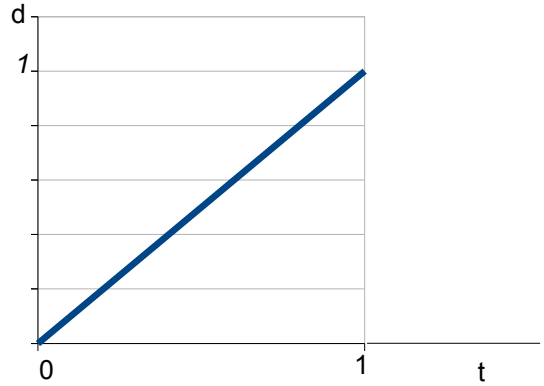
$$d = \int \sin(t)\,dt = -\cos(x) \tag{3.11}$$

$$d = \frac{-\cos(t\pi)+1}{2} \tag{3.12}$$



Obr. 3.13 Převodní funkce pro sinus

Takto lze implementovat libovolné funkce, u kterých lze vypočítat jejich integrál. I u předem zmíněné funkce s konstantní rychlosti 3.11. lze snadno vypočítat její převodní funkci, kdy po převedení do rozsahu <0,1> dostaneme vztah d=t a tedy že současná pozice zůstane nezměněná. Graf je zobrazen na obrázku 3.14.



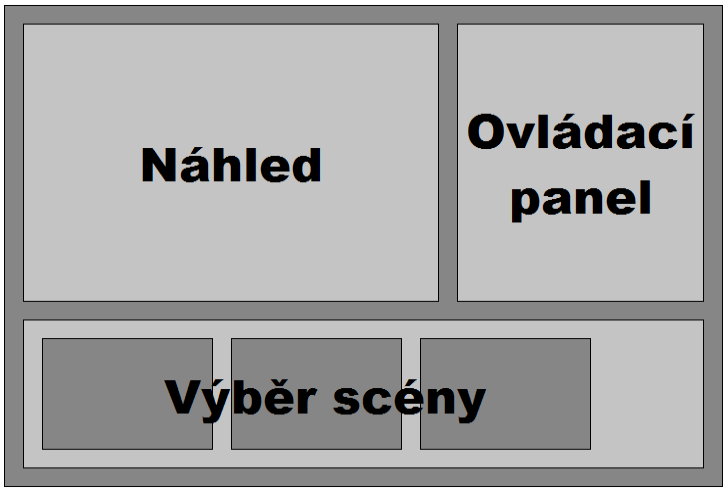
Obr. 3.14 Převodní funkce konstantní rychlosti

3.6 Uživatelské rozhraní

Cílem diplomové práce je vytvořit editor pro tvorbu sekvencí a následné přehrávání. Výsledkem práce by tedy měl být editor, který bude pro uživatele jednoduše ovladatelný. Uživatel by se měl

rychle zorientovat ve způsobu práce s editorem a tvorba obrazových sekvencí by měla být co nejjednodušší.

Aby se zkrátila doba po kterou se uživatelé seznamují s editorem je vhodné nevymýšlet nový koncept ovládání, ale využít zkušenosti uživatelů s jinými editory. Může se jednat například o nástroje pro střih videa, Kde se ve spodní části obrazovky nachází část výběru jednotlivých video-sekvencí, což by se v našem případě nahradilo výběrem scény. A zbytek obrazovky byl rozdělen na část s náhledem a část s nabídkou pro editaci, přehrávání a další nastavení. Toto řešení pomocí grafického uživatelského rozhraní může být poměrně robustní a má význam zejména při rozšiřování funkcionality. Takový editor je nejvhodnější ovládat s pomocí myší. Rozhraní může být rovněž vhodně doplněné o klávesové zkratky. S jejich pomocí lze některé funkce provádět snadněji. Návrh rozhraní je zobrazen na obrázku 3.15.



Obr. 3.15 Návrh uživatelského rozhraní

Při tvorbě grafického uživatelského rozhraní je nutné také vzít v úvahu, zda využít knihovny s vytvořenými prvky, nebo si prvky jako tlačítka, posuvníky implementovat. Výhodou využití knihovny je snadnější tvorba uživatelského rozhraní, naopak výhodou implementace vlastního rozhraní je možnost vytvářet uživatelské rozhraní přesně podle požadavků aplikace a nejsme omezeni možnostmi použité knihovny. Předpokládá se, že se bude návrh uživatelského rozhraní v průběhu implementace měnit.

3.7 Rozšíření zadání

Zadání práce není příliš podrobné. Proto je nutné toto zadání rozšířit o další možnosti. Zejména o funkce editoru. Zde jsou uvedeny možnosti, jak by bylo možné editor dále rozvíjet. Podrobněji jsou rozšíření popsány v části implementace.

3.7.1 Postprocessing efekty

Na výsledný obraz je možné aplikovat různé obrazové filtry, které můžou změnit celkový dojem z animace. Tyto postprocessing efekty je vhodné implementovat s pomocí fragment shaderů grafické karty. Například je možné změnit scénu na černobílou, kdy se ve fragment shaderu původní barva převede na barvu černobílou, stejným způsobem lze zabarvit scénu do sépiových barev. Je možné využít i pokročilejší efekty, jako historický film, kdy by byl obraz převeden do jiných barev například sépiových a následně by do výsledného obrazu byly vloženy různé vady, které se u starých filmů vyskytují. Lze navrhnout více možných efektů. Implementované efekty jsou popsány v části implementace.

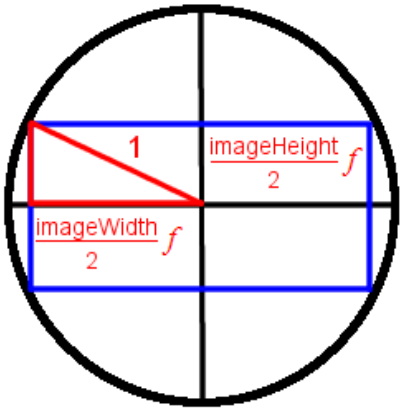
3.7.2 Přechody

Cílem by mělo být vytvoření funkcí pro efektový střih jednotlivých scén v prezentaci. Knihovna nemusí být příliš rozsáhlá, dostačující budou pouze základní funkce s nástinem dalšího rozšiřování. Pro vytvoření přechodů je možné využít míchací funkci knihovny OpenGL. Kdy je možné podle alfa kanálu nastavit průhlednost obou scén. Nejdříve se tedy zobrazí první scéna a následně se přes ni zobrazí scéna druhá s upraveným alfa-kanálem. To umožní provádět různé efekty s prolínáním obrazu. Další možností je využít stencil buffer, kterému se nastaví, které části obrazu se zobrazí. To umožní využít efekty jako setření, kdy na počátku je zobrazena první scéna a poté se bude postupně maskovat první scéna a zobrazovat scéna druhá. Jedná se pouze o nástin způsobu, jak řešit přechody a příklady jsou pouze pro demonstraci. Implementované efekty jsou popsány v části implementace.

4 Implementace

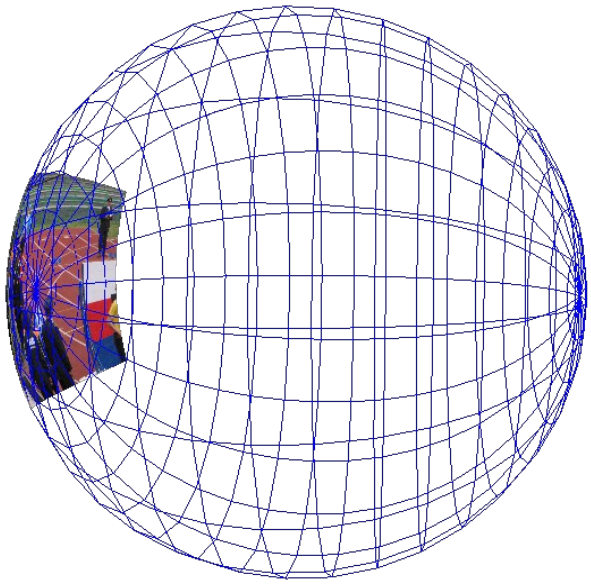
4.1 Transformace 2D do 3D

Transformace 2D souřadnic do 3D prostoru nebylo nutné implementovat. Byla využita knihovna pro převod těchto souřadnic. Autorem této knihovny je vedoucí práce Michal Seeman. Z této knihovny byla využita funkce FilmToAngleVector. Ta vypočítá pozici bodu ve 3D prostoru z 2D souřadnice fotografie a ohniskové vzdálenosti, se kterou je fotografie vyfotografována. Závisí také na typu optické soustavy fotoaparátu. Rozměry fotografie je nutné změnit tak, aby byla obsažena v kruhu, jehož poloměr je 1 a tedy aby byly body snímku maximálně ve vzdálenosti 1. Výpočet je znázorněn na obrázku 4.1, kde je nutné vypočítat faktor zmenšení *f*, kterým vynásobíme rozměry fotografie a získáme tak snímek požadovaných rozměrů. Hodnotu veličiny *f* lze vypočítat vzorcem 4.1. V knihovně je implementováno několik typů čoček pro různé typy objektivů. Je tedy nutné kromě zadané ohniskové vzdálenosti a pozice bodu zvolit vhodný typ čočky. Pokud přesně není známá charakteristika fotoaparátu je nejvhodnější zvolit ideální tenkou čočku TIdealFlatLens, protože výrobci fotoaparátů se ve většině případu snaží tomuto typu čočky přiblížit. Na obrázku 4.2 je zobrazeno umístění fotografie do scény a její namapování na kouli.



Obr. 4.1 Znázornění faktoru zmenšení *f*

$$f = \sqrt{\frac{4}{imageWidth^2 + imageHeight^2}} \tag{4.1}$$



Obr. 4.2 Umístění fotografie do scény

4.2 Zobrazení scény

Při zobrazení scény je textura zdrojové fotografie namapována na polygony. Fotografie je rozdělena mřížkou na několik částí. Rozdělení se provede takovým způsobem, že šířku snímku rozdělíme na M zadaných částí na řádku a N zadaných částí ve sloupci. Výsledný obraz tedy bude mřížka MxN polygonů. Postupujeme z levého horního rohu a postupně přičítáme hodnotu šířky a výšky jedné části. Tím získáváme jednotlivé body mřížky, které přepočítáváme pomocí dříve uvedené funkce FilmToAngleVector. Výpočet se provede ze zadané souřadnice snímku, typu optické soustavy a ohniskové vzdálenosti objektivu. Takto jsou vypočítány všechny vrcholy jednotlivých čtyřúhelníků mřížky. Výsledné čtyři vrcholy udávají pozici polygonu v prostoru a tento polygon je zobrazen s texturou vybrané části. Texturovací souřadnice odpovídají souřadnicím jednotlivých částí textury. Takto se provede výpočet pro celou scénu a výsledek je zobrazen. Požadované parametry snímku a optické soustavy jsou načteny z právě aktivní scény.

Při implementaci nastaly problémy s velkými obrázky, kdy OpenGL neumí pracovat s texturami s vyšším rozlišením. Na počítači kde byla prováděna implementace byl maximální možný rozměr textury 2048 bodů. Proto bylo nutné obraz rozdělit na několik menších obrazů, ze kterých jsou vytvořeny textury se kterými je již knihovna OpenGL schopna pracovat. Každý z těchto podobrazů je rozdělen mřížkou M,N. Výpočet texturovacích souřadnic je stejný jako v původní variantě s tím, že souřadnice odpovídají zvolenému podsnímku. Výpočet souřadnic však musel být upraven, aby pozice neodpovídala pozici v podsnímku, ale pozici v původním velkém obraze. Způsob počítání je stejný. Od počáteční pozice se posunujeme o zadaný krok. Pouze se změnil vzorec, kterým

toto vypočítáme. Ve vztahu 4.2 je uvedeno počítání počáteční pozice ve vodorovné ose a ve vztahu 4.3 je uvedeno vypočítání kroku. V těchto vzorcích je *f* faktor zmenšení se vzorce 4.1, *imageWidth* je počet pixelů obrazu, *textureWidth* je maximální rozměr textury a *c* označuje, o kolikátou texturu na řádku se jedná. Pokud je rozměr obrazu menší než maximální rozměr textury, pak *c* je rovno 0. Pro počítání ve svislé ose se pouze zamění šířka za výšku.

$$pos = \frac{f \cdot imageWidth}{2} - f \cdot c \cdot textureWidth \tag{4.2}$$

$$step = \frac{f \cdot textureWidth}{M} \tag{4.3}$$

Implementace dovoluje měnit počet polygonů ze kterých je scéna vytvořena. Protože je jedna strana textury 2048 bodů, je pro snadný výpočet počet polygonů zarovnán na mocniny dvojky. To umožňuje měnit kvalitu zobrazení v závislosti na výkonu počítače. Počet polygonů by se měl zvolit takový, aby nedocházelo k deformaci obrazu. U vysokého počtu polygonů již neroste kvalita zobrazení, ale pouze výpočetní náročnost. Vhodný počet polygonů je $\{2^5,2^6,2^7\}$. Vyšší počet polygonů již nemá význam. Pro texturu o rozměrech 2048x2048 a kvalitu 7 (2^7) je počet polygonů scény 128x128=16 384 polygonů.

4.3 Kamera

V reálné implementaci se nastavení kamery provede pomocí projekční matice a modelview matice. V návrhu byly popsány dvě varianty implementace operace zoom. Implementovány byly obě varianty, ale byla zvolena varianta se změnou zorného úhlu. Projekční matice se nastaví pomocí funkce gluPerspective, které je zadán požadovaný obrazový úhel. Výpočet úhlu je proveden podle vztahu 3.7. Protože je funkci gluPerspective zadán úhel na ose y, je ve výpočtu nahrazena šířka 36mm výškou kinofilmu 24mm.

Následně se modelview maticí nastaví pozice kamery pomocí postupného provedení jednotlivých transformací. Pořadí jednotlivých rotací je zdůvodněno v kapitole 3.2.1.

```
glRotatef(rotace.z, 0.0, 0.0, 1.0); // rotace kolem z
glRotatef(rotace.x, 1.0, 0.0, 0.0); // rotace kolem x
glRotatef(rotace.y, 0.0, 1.0, 0.0); // rotace kolem y
```

Ovládání kamery je v současné době prováděno pomocí myši a to tím způsobem, že po stisku levého tlačítka myši a tažením vlevo a vpravo se nastavuje otáčení ve vodorovném směru, tažením nahoru a dolů ve svislém směru. Provádění operace zoom je prováděno pomocí otáčení kolečka myši nahoru a dolů. Po stisku pravého tlačítka myši a tažením vlevo a vpravo se nastavuje rotace kolem

osy kamery. Při této rotaci se však otáčí osy x a y souřadného systému a při následném posunutí kamery v horizontálním či vertikálním směru se kamera posunuje v šikmém směru v závislosti na natočení. Abychom tento jev eliminovali je nutné aby ovládání kamery sledovalo natáčení kolem osy kamery a přepočítávalo zadané posunutí v závislosti na rotaci. Pro přepočet lze využít vztahu 2.3 uvedeného v kapitole 2.5.2.

4.4 Práce s informacemi

Během implementace bylo nutné upravit a doplnit informace do struktury uvedené v kapitole 3.3. Protože se při vykreslování pracuje s texturami, které se nanáší na polygony umístěné v prostoru, bylo vhodné do struktury uložit kromě obrázků i odkazy na textury v paměti, které z obrázků byly vytvořeny. Při vykreslování již není nutné tuto texturu vytvářet a je pouze připojena k vykreslovaným polygonům. Tento způsob umožňuje rychlejší vykreslování a současně jednoduchý přístup ke všem texturám, kdy každá scéna si uchovává informace o své textuře. Uvedením textur mezi informacemi ve scéně se mírně porušuje předchozí vlastnost, kdy struktura pro uložení informací není závislá na konkrétní grafické knihovně. Protože ostatní části jsou psány pod grafickou knihovnou OpenGL, tak v případě změny grafické knihovny by se ostatní části musely předělávat. Naopak změna typu ukládané textury OpenGL na jiný typ textury jiné knihovny je možné provést bez větších změn. Tuto drobnou nevýhodu tedy není nutné brát v úvahu a textury ve struktuře ponechat.

Jak již bylo zmíněno dříve, během implementace nastal problém s texturami ve vysokém rozlišení. Ty bylo nutné rozdělit na několik menších. Ve struktuře bylo upraveno uložení textury. Původně uložená textura v informacích o scéně byla změněna na seznam textur a současně byly přidány informace důležité pro vykreslování a správné umístění textur ve výsledné scéně. Bylo přidáno rozlišení rozdělených obrazů a jejich počet na řádku a ve sloupci.

Pro výpočet dráhy kamery jsou využívány křivky. Aby se nemusela dráha kamery přepočítávat ze zadaných bodů, je výsledná dráha uložena také mezi informacemi o scéně. Tím se ušetří výpočetní výkon. Při požadavku na nastavení kamery na pozici je na křivce průběhu tento bod nalezen a zjištěna jeho hodnota. Pak je tedy křivka přepočítána pouze v případě přidání či odebrání některého z řídících bodů nebo v případě změny typu křivky. Současně je s vypočítanou křivkou uložen i její typ.

Přestože všechny operace pracující se strukturou informací jsou vytvořeny tak, aby výsledná struktura zůstala konzistentní, může nastat nekonzistence při načítání ze souboru. Načítání předpokládá, že soubor obsahuje konzistentní informace. Pokud bude soubor vytvořen editorem, je jeho konzistence zaručena. Pokud by však uživatel neodborně upravoval soubor s projektem sám, mohl by do něj zanést nekonzistenci. Tím že není nutné testovat konzistenci, je načtení mnohem

jednodušší. Metody pro práci se strukturou informace provádí při úpravách další přepočty. Při načítání souboru se operace jako přidávání stále opakují. Bylo by nutné v každém kroku přidávání provádět další výpočty jako přepočet dráhy kamery. Vyřazením těchto testů a výpočtů dosáhneme rychlejšího načtení.

Do struktury přibyla ještě položka udávající aktivní prvek. V případě globálních informací byla přidána hodnota udávající aktuální scénu. V případě změny nastavení se pak pracuje s tou scénou, která je právě aktivní. Stejně jako je ukládáno číslo aktivní scény tak je pro každou scénu ukládáno číslo aktivní pozice, které udává, se kterým nastavením pozice kamery se právě pracuje. Pro obě varianty pak všechny metody pracují s aktivním prvkem, pokud není zadán jiný. Aktuální prvek nemusí být vybrán. V tom případě má položka hodnotu nedefinovaného prvku.

4.5 Optimalizace rychlosti

Kromě většího množství drobných optimalizaci byla největší změna provedena v části zobrazení scény. Vždy při vykreslení scény byly znovu počítány jednotlivé souřadnice polygonů v prostoru. Aby bylo dosaženo zrychlení, byly jednotlivé vrcholy a texturovací souřadnice vypočítány dopředu. Vrcholy jsou uloženy v seznamu vrcholů a souřadnice v seznamu souřadnic. Při vykreslování jsou pouze ze seznamů načítány vrcholy a souřadnice. Tím lze dosáhnout vyšší rychlosti zobrazení. Po implementaci této optimalizace na testovací sestavě mohl být zvýšen počet zobrazovaných polygonů více než 4x se zachováním původní rychlosti vykreslování. Zvýšení není měřeno nějakou exaktní metodou, ale bylo zjištěno na základě pozorování. Použitím této optimalizace však musíme v případě změny parametrů scény zajistit přepočítání obou seznamů.

Musíme ale také vzít v úvahu to, že každá scéna má jiné parametry. Proto by bylo nutné při každé změně scény přepočítávat oba seznamy. Stále by se jednalo o zvýšení rychlosti, protože scéna by se nepřepočítávala při každém vykreslení, ale vždy pouze jednou. Aby bylo možné i toto přepočítávání eliminovat, byl ke každé scéně připojen objekt, který se stará o její zobrazování. Každá scéna má tedy svůj vlastní seznam podle kterého je vykreslována. Přepočítání se provede pouze při změně parametrů konkrétní scény. Předpoklad je, že tyto parametry optické soustavy budou na počátku nastaveny a již se dále měnit nebudou. Proto se výpočet provede jednou. Stále je však možné tyto parametry měnit, je nutné však dále zajistit přepočítání.

Toto řešení také umožňuje přenést textury dříve uložené mezi informacemi o scéně do třídy pro zobrazování. Při vykreslování jsou textury lépe přístupné a druhou výhodou je to, že funkce pro práci s texturami, které jsou závislé na konkrétní grafické knihovně odstraníme ze struktury. Ta bude tedy opět nezávislá na konkrétní grafické knihovně. Při případné změně grafické knihovny bude struktura pro uložení opět funkční bezezměny. Pouze by bylo nutné změnit třídu pro zobrazování.

Současně s přesunutím kreslící funkce do informací o struktuře, byla přenesena i informace o kameře. Původně se pracovalo s jednou kamerou, které se nastavovala pozice dle aktivní scény. Po přesunutí již má každá ze scén svou vlastní kameru. Tento způsob nepřináší žádné zrychlení. Výhodou tohoto řešení je stejný způsob práce s kamerou jako s nyní přesunutým vykreslením scény. Při zobrazování je tedy nejprve nastavena kamera konkrétní scény a následně je tato scéna vykreslena. To umožňuje snadné vykreslování, kdy je zavolána pouze jediná metoda pro vykreslení scény. Drobnou výhodou může ještě být, že při změně kamery v jedné scéně zůstává nastavení v ostatních scénách zachováno.

4.6 Průběh rychlosti pohybu

Změna rychlosti kamery byla provedena podle uvedeného návrhu. Implementovány však byly dvě varianty, kdy se jednotlivé varianty liší podle vytyčení počátečního a koncového bodu průběhu. Každý z režimů jiným způsobem vypočítá normalizovaný čas na ose x. V prvním režimu lokálním jsou první a druhý bod vždy sousední pozice kamery zadané uživatelem a každý z úseků je popsán vlastní funkcí, která definuje její průběh rychlosti pohybu. V tomto režimu je normalizovaný čas vypočten tak, že je od aktuálního času ve scéně odečtena hodnota času výskytu na počáteční pozici čímž získáme relativní čas vzhledem k počátečnímu bodu. Tento čas vydělíme zadanou dobou přechodu z počátečního do koncového bodu a dostaneme souřadnici na ose x charakteristiky a získanou hodnotu na ose y opačným postupem převedeme na novou pozici kamery.

V druhém režimu je jednou charakteristikou popsán pohyb kamery v jedné scéně. Tento režim je globální. Počátečním bodem je první bod scény a koncovým bodem poslední bod scény. Charakteristiky mohou být stejné v obou případech. V druhém režimu se tato charakteristika roztáhne na celý pohyb ve scéně. Tento režim se liší pouze výpočtem normalizované souřadnice na ose x průběhu a zpětný přepočet pro nastavení pozice. Aktuální čas ve scéně je vydělen dobou výskytu koncového bodu. V tomto režimu může kamera procházet určitými úseky vyšší rychlostí než jinými. Proto v tomto režimu nemusí kamera procházet zadanými vnitřními úseky v zadaném čase, ale může se na zadané pozici objevit dříve nebo později. Uživatelem zadaná doba výskytu na určité pozici ovlivní skutečnou dobu výskytu na této pozici při přehrávání, ale skutečná a zadaná doba nemusí být ekvivalentní.

Protože jsou jednotlivé dráhy pohybu kamery rozděleny na jednotlivé složky rotace a přiblížení, je i rychlost kamery rozdělena na tyto složky. Každá ze složek může mít definovánu vlastní charakteristiku a při výpočtu se počítá charakteristika pro každou ze složek zvlášť.

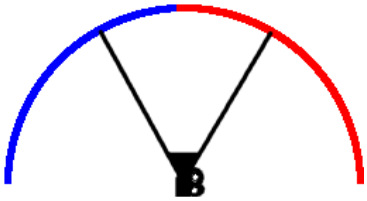
4.7 Střihové efekty

Jak již bylo zmíněno dříve, byly implementovány jednoduché střihové efekty s využitím blendingu a stencil bufferu. Každá z částí pracuje jiným způsobem, ale obě využívají přechodovou hodnotu, která udává, v jaké pozici se právě nachází. Přechodová hodnota se vypočítá podle vztahu 4.4. LocalTime udává aktuální pozici ve scéně, sceneLenght je celková doba trvání scény a effectLenght je nastavená délka přechodu. Přechodový efekt nastává, pokud je přechodová hodnota v intervalu (0,1>. Pokud je menší nebo rovna nule, přechod ještě nenastal. Pokud je hodnota rovna 1 byl přechod dokončen a dále běží pouze druhá scéna. Poté je na základě vypočítané přechodové hodnoty nastaven efekt s maskováním nebo prolnutím v závislosti na zadaném typu. Oba typy je také možné zkombinovat.

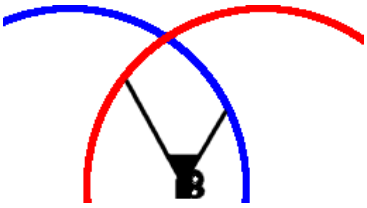
$$transitionValue = \frac{localTime - sceneLenght + effectLenght}{effectLenght} \tag{4.4}$$

Během vykreslování přechodů jsou přes sebe zobrazeny dvě scény. Obě jsou tedy namapovány na jednotkovou kouli. V ideálním případě by se druhá zobrazovaná scéna nacházela na stejném místě jako scéna první a proto by již nemusela být zobrazena. V reálné implementaci je scéna složena z jednotlivých vrcholu a vrcholy polygonů obou scén se nemusí nacházet na stejném místě v důsledku zaokrouhlování a aproximace kulové plochy rovinnými plochami. Některé vrcholy druhé scény by se tedy mohli nacházet před a některé za první scénou. Proto je nutné během přechodů vypnout testování depth bufferu. Tím bude druhá scéna vykreslena nezávisle na konkrétní pozici jednotlivých polygonů. Protože není třeba řešit viditelnost pomocí depth bufferu při zobrazování kulové plochy, může zůstat vypnutý trvale. Nejen během přechodů.

V editorech pro střih video-sekvencí je možné pracovat ještě s jedním typem přechodu a to s takovým, kdy jedna scéna je odsunuta scénou druhou. Tento typ přechodu je vhodný pro 2D video sekvence, ale u trojrozměrného promítání je tento způsob nevhodný. Virtuální kamera zobrazuje scénu tak, jako kdyby uživatel stál uprostřed jak je znázorněno na obrázku 4.3. Pokud bychom použili posunutí celé scény, pak by se pozorovatel nacházel ve dvou středech a tedy na dvou místech současně. Po posunutí by tedy došlo k chybě při zobrazení, která je znázorněna na obrázku 4.4. Uživatel při sledování posunutých scén není ve středu promítací koule a dochází k deformaci obrazu. Jedinou možností je provést odsunutí pohybem kamery a to lze provést za použití maskování. Současně s průběhem maskovací funkce bude nastaven pohyb kamery do strany. Tím dosáhneme podobného efektu jako předem uvedené posunutí ve 2D editorech pro střih. V tomto případě však pozorovatel u obou scén zůstane ve stejném místě. Pouze jsou z jedné pozice viditelné dvě různé scény.



Obr. 4.3 Správné zobrazení scén – maskování



Obr. 4.4 Chybné zobrazení scén – posunutí

4.7.1 Prolnutí

Efekt prolnutí první scény do druhé se vytvoří s pomocí blendingu. Vytváří se při vykreslování scény. Nejprve je vykreslena první scéna normálním způsobem a následně je vykreslena scéna druhá s nastaveným alfa-kanálem. Hodnotou alfa-kanálu je vypočítaná přechodová hodnota. Alfa kanál je ve scéně nastaven pomocí míchací barvy, které je nastavena hodnota *glBlendColor(1.0f, 1.0f, 1.0f, alpha)* a následně je nastavena míchací funkce *glBlendFunc(GL_CONSTANT_ALPHA, GL_ONE_MINUS_CONSTANT_ALPHA)*. Tato scéna je vykreslena se zapnutým blendingem přes scénu původní.

Na rozdíl od efektů využívajících stencil buffer nelze u tohoto přechodu vytvářet více různých typů přechodů. Tento přechod je možné pouze vypnout nebo zapnout. Přechod je zobrazen na obrázku 4.5.

Obr. 4.5 Prolnutí obrazů

4.7.2 Maskování

Maskováním lze vytvářet takové efekty, kdy na části obrazovky se zobrazí první scéna a na zbývající části scéna druhá. Pokud se bude maskovaná část měnit v závislosti na čase přechodu, dosáhneme zajímavých přechodů. Stejně jako v předchozím případě, kde se vykreslila první scéna a přes ni následně druhá, i zde je možné vykreslit první scénu celou a pouze u vykreslení druhé scény

vybrat požadovanou oblast a pouze tuto překreslit. U první scény je tedy nastavena funkce *glStencilFunc(GL_ALWAYS, 0x1, 0x1)*, která zaručí že bude vždy celá vykreslena. Protože maskování pracuje s výsledným obrazem a ne s jednotlivými scénami, je nutné změnit perspektivní projekci kterou jsou vytvářeny scény na projekci ortogonální.

Následně je nutné nastavit hodnoty stencil bufferu. Pro každý bod, kde se bude zobrazovat druhá scéna je nastavena jeho hodnota na 0x1. Nastavení se provede pomocí *glStencilOp(GL_REPLACE, GL_REPLACE, GL_REPLACE)*. Ještě je nutné zakázat zápis do color bufferu *glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE)*.

Nyní je již možné vykreslit libovolný polygon, který bude udávat místa s druhou scénou. Může zde být zobrazen jednoduchý obdélník, jehož velikost se bude postupně měnit a se zvyšováním přechodové hodnoty se bude zvětšovat. Tento výsledný efekt je znázorněn na obrázku 4.6. Vykreslení tohoto obdélníku se provede pouze zadáním čtyř vektorů polygonu.

```
glVertex2f (0.0, 0.0);
glVertex2f (0.0, transitionValue*maxHeight);
glVertex2f (transitionValue*maxWidth, transitionValue*maxHeight);
glVertex2f (transitionValue*maxWidth, 0.0);
```

Přidání dalších efektů lze jednoduše provést přidáním další funkce do třídy efektů a do seznamu v uživatelském rozhraní. Každá přidaná funkce slouží pouze pro vykreslení jednotlivých objektů do stencil bufferu. Inicializaci a dokončení provede třída automaticky. V současné době jsou implementovány pouze základní přechody. Ty vychází ze zde uvedeného příkladu. Uvedená varianta obdélníka, který se postupně zvětšuje z levého horního rohu až po pravý dolní roh je upravena na varianty pouze v jednom směru a to zleva doprava a shora dolů. Dále je implementovaná inverzní varianta přechodu zprava doleva. U inverzní varianty zůstává koncový bod na maximální hodnotě a mění se pouze počáteční a to podle vztahu *(1-transitionValue)*maxWidth,* kde se počáteční bod přesouvá směrem k začátku.

Po vykreslení obsahu bufferu je vypnut zápis *glStencilOp (GL_KEEP, GL_KEEP, GL_KEEP)*, je povolen zápis do color bufferu *glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE)* a je nastavena funkce stencil bufferu *glStencilFunc(GL_EQUAL, 0x1, 0x1)*, aby vykreslovala pouze na pozicích s 0x1. Po nastavení se vykreslí druhá scéna přes první.

Obr. 4.6 Maskování dvou scén

4.8 Postprocessing efekty

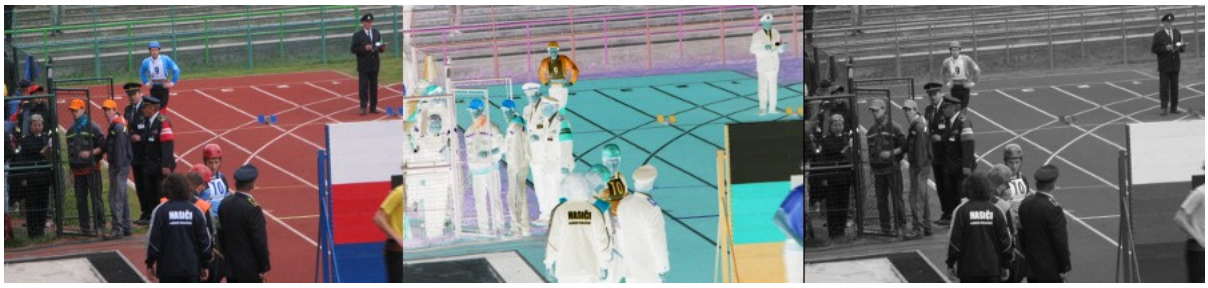
Jak již bylo zmíněno v návrhu, tyto efekty využívají fragment shaderu grafické karty. Slouží pro změnu výsledného obrazu, kdy každý výsledný pixel je vypočítán podle zadaného vzorce. Možnosti výsledného efektu jsou omezeny pouze možnostmi pixel shaderů grafické karty. Pro přidání dalšího efektu stačí napsat program, který zadanou činnost provede a tento program připojit k ostatním. Dále je pak nutné přidat další položku pro tento program do seznamu efektů v uživatelském rozhraní.

Všechny napsané programy pro shadery je před použitím nutné zkompilovat. V tomto případě jsou dvě možnosti a to zkompilovat program pouze v případě jeho potřeby, nebo zkompilovat všechny programy již během inicializace. Zvolena byla varianta během inicializace a všechny programy jsou následně dostupné vždy v případě potřeby. Tím se mírně prodlouží doba inicializace, ale dále již není uživatel zdržován dodatečnou kompilací. V případě požadavku na aktivaci vytvořeného programu, je vrácen ukazatel tohoto programu a tento je připojen ke grafické kartě jako program pracující s fragment shadery. Vykreslení se provede takovým způsobem, že před vykreslením každé scény se aktivuje shader program provádějící zadaný efekt. Tím je na všechny pixely scény aplikován program a výsledek je zobrazen.

Tím že byly využity fragment shadery, je nutné pro správnou funkci využít OpenGL alespoň ve verzi 2.0. Současně je nutné vzít v úvahu, že použité programy mohou být výpočetně náročnější než je původní vykreslovací řetězec. Výrazným způsobem mohou snížit rychlost přehrávání. Proto je nutné při volbě efektů brát v úvahu také dostupné možnosti grafické karty. Problémem může být přenášení projektů mezi různými počítači, kdy na jednom počítači s výkonnou grafickou kartou běží přehrávání dostatečně rychle, ale na jiné pomalejší grafické kartě dochází ke zpomalení. V tomto případě je nutné vypnout výpočetně náročné programy nebo snížit počet snímků za vteřinu a tedy zhoršit kvalitu přehrávání.

Byly implementovány pouze základní efekty, které demonstrují možnosti dalšího rozšiřování. Mezi základní efekty patří negativ a černobílý obraz. Výstupem z fragment shaderu je barva pixelu RGB. Pokud není žádný efekt zvolen, je pouze pixel ze vstupu přenesen na výstup. V případě, že je zvolen negativ je původní barva pixelu (R,G,B) na vstupu změněna na ((1-R),(1-G),(1-B)) a tento výsledek udává barvu výsledného pixelu. U černobílého obrazu je převeden původní barevný obraz do šedo-tónového. Nejprve je vypočítána intenzita podle vztahu 4.5. a výsledná barva ve stupních šedi je definována intenzitou (I,I,I). Jedná se o jednoduché efekty, které nemají velký vliv na rychlost vykreslování a jejich rychlost je podobná jako rychlost vykreslování s vypnutými efekty. Výsledné efekty jsou zobrazeny na obrázku 4.7.

$$I = 0.299\mathrm{R} + 0.587\mathrm{G} + 0.144\mathrm{B} \tag{4.5}$$



Obr. 4.7 Obrazové efekty (bez efektu, negativ, stupně šedi)

4.9 Animační engine

Tímto názvem není myšlen engine, jak ho známe z her a ostatních nástrojů, kde slouží pro zobrazování komplexních scén. V tomto případě se pouze jedná o nástroj, který se stará o správné zobrazení scén, přepnutí na další scény, přechody mezi scénami a zejména o výpočet dráhy kamery během přehrávání animací. Spolupracuje s časovačem a poskytuje základní informace o průběhu animace. Oznamuje, zda právě běží animace. Pokud animace běží, tak poskytuje informace o jejím průběhu.

Před startem animace je provedena počáteční inicializace. Následně časovač v pravidelných časových intervalech aktualizuje engine, který nastaví kamery do pozice odpovídající aktuálnímu času animace a případně řeší přechod na další scény. Po aktualizaci nastavení je inkrementován čas animace. Animace nepracuje s celkovým časem, ale pouze s lokálním časem trvání konkrétní scény.

Pro nastavení pozice kamery je lokální čas scény převeden do intervalu <0,1> se kterým pracují funkce pro výpočet dráhy kamery a pro výpočet průběhu rychlosti. Dle zadané funkce průběhu rychlosti je upravena vypočítaná hodnota, aby odpovídala zadanému průběhu. Převod je popsán v kapitole 3.5. Dále je získaná hodnota předána na vstup funkce pro výpočet dráhy kamery a ta vrací požadované nastavení kamery. Výpočet dráhy je popsán v kapitole 3.4. Pro aktuální scénu je nastavena kamera dle získaného nastavení.

Po nastavení kamery pro aktuální scénu je testováno, zda již nenastane přechod z aktuální scény do další. V případě že má nastat přechod, vypočítá se přechodová hodnota v intervalu <0,1>. Následně je nastavena pozice kamery stejným způsobem jako u kamery první. Pouze je přepočítán lokální čas první scény na lokální čas druhé scény. Po skončení přechodu se stává druhá scéna aktivní a výpočet se opakuje dokud neskončí scéna poslední.

Při vykreslení během animace je nejprve nastavena kamera první scény. Poté vykreslena první scéna a dále v závislosti na přítomnosti přechodu je případně nastavena kamera druhé scény a vykreslena druhá scéna se zadaným maskováním a průhledností. Přechody jsou popsány v kapitole 4.7. Před vykreslením scén jsou případně ještě zvoleny programy pro obrazové efekty popsané v kapitole 4.8.

Po skončení animace jsou nastavení enginu resetována. Je zrušena informace o běžící animaci a časovač je zastaven. Animaci je možné během přehrávání zapauzovat, kdy nemusíme řešit žádné nastavení, pouze stačí vypnout časovač. Ten nebude dále dodávat impulzy pro postup v animaci a ta tedy nebude pokračovat dokud nebude časovač opět spuštěn. Během přehrávání animace je vypnuta možnost editace. Proto i při zapauzování je tato možnost nedostupná a editace bude zpřístupněna pouze při skončení nebo zastavení animace.

Původně bylo přehrávání animace s frekvencí 25 snímků za sekundu. U úseků s rychlým pohybem kamery docházelo k jemnému trhání obrazu. Proto byl počet snímků zvýšen na 50 za sekundu. Aby však bylo umožněno přehrávání i na pomalejších systémech, které by 50 snímků za sekundu nemuseli zvládat, byla zachována možnost zobrazovat 25 snímků. To lze provést s pomocí časovače, kterému lze nastavit prodlevu 20ms pro 50FPS, nebo 40ms pro 25FPS. To by však byla rychlost přehrávání v obou případech jiná. Proto v závislosti na zvolené frekvenci snímků je jiná aktualizace času animace. Čas animace je celé číslo a postup animací se provádí přičítáním jedničky. Pokud budeme v případě 25 snímků přičítat dvojku, pak s poloviční frekvencí budeme pracovat pouze s polovinou bodů animace. Tím dosáhneme v obou případech stejné rychlosti přehrávání. Takto by bylo možné využít i více různých frekvencí, ale průběh animace při 50 snímcích je dostatečně jemný a oko by při této frekvenci nemělo žádné trhání pozorovat.

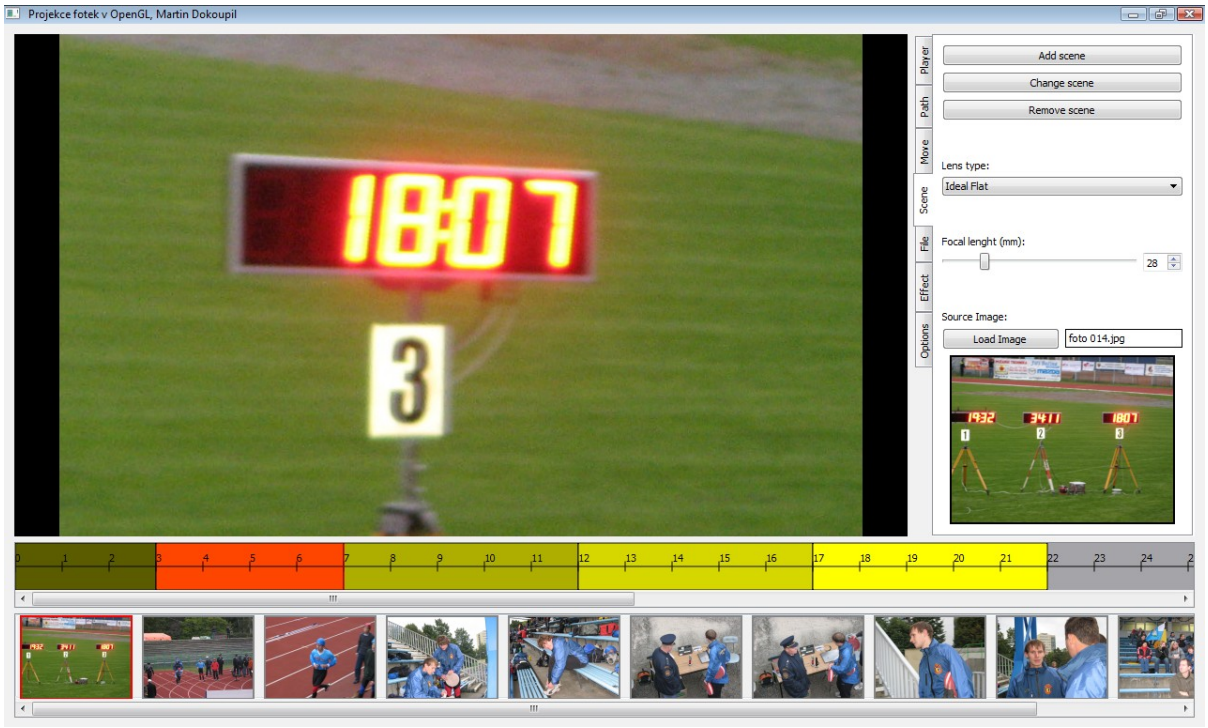
4.10 Uživatelské rozhraní

U prvotní verze uživatelského rozhraní byly vytvořeny vlastní ovládací prvky, které byly postavené na rozšíření OpenGL GLUT. Byly vytvořeny jednoduchá tlačítka, seznamy a některé další prvky, které umožnily základní práci s uživatelským rozhraním. Při dalším rozšiřování možností uživatelského rozhraní začaly být možnosti jednotlivých prvků a počet potřebných prvků časově náročné na implementaci. Pokud by se pokračovalo s vlastním uživatelským rozhraním, stále by rozhraní umožňovalo stejné funkce jako výsledná verze, ale některé prvky rozhraní by se neovládali s takovým komfortem jako prvky využité ve výsledné verzi.

Z těchto důvodů byla zvolena varianta využití již vytvořené knihovny pro tvorbu uživatelského rozhraní. Požadavky na rozhraní byly podpora OpenGL, multiplatformní, kvalitní dokumentace a dostatečná podpora od tvůrců, jednoduchá instalace a práce s rozhraním. Jako vhodný kandidát bylo zvoleno rozhraní Qt. To splňuje dané požadavky a umožňuje snadnou práci s rozhraním, které má velké možnosti. To také proto, že za ním stojí velká firma starající se o jeho podporu. Převedení původního rozhraní na nové Qt bylo nenáročné. Pouze u speciálních prvků ovládání muselo dojít k větší úpravě. Po implementaci a otestování nového rozhraní bylo zjištěno, že původní vlastní rozhraní bylo mírně rychlejší. Drobné zpomalení ale nebylo nijak výrazné. Projevovalo se spíše u přehrávání.

Drobné zpomalení bylo odstraněno zrušením aktualizací obsahu prvků při přehrávání, kdy je veškerá editace zakázaná. Výhody nového rozhraní převyšují všechny drobné zápory, které implementace s využitím Qt má.

Jak již bylo uvedeno v návrhu, uživatelské rozhraní je složené z několika částí. Výsledné uživatelské rozhraní je na obrázku 4.8. Ve spodní části je seznam všech scén v projektu, které jsou znázorněny obrázky. Nad seznamem scén je plánovač s časovou osou, který slouží k nastavení pozice kamery jedné scény v průběhu animace. Dále je v pravé části panel s veškerými ovládacími prvky sloužící pro práci s animací, efekty, ukládání a načítání projektů a nastavení přehrávání animací. V levé části se zbylým místem se nachází náhled na scénu. Přehrávání animací je možné jak v náhledu tak v režimu přes celou obrazovku.



Obr. 4.8 Uživatelské rozhraní editoru

Během testovaní byl zjištěn problém při změně velikosti okna s náhledem a při přepínání do režimu přes celou obrazovku. Po zvětšení docházelo k chybě, kdy se část obrazu nepřekreslovala. Podobný problém se však vyskytoval i u oficiálních ukázkových příkladů a proto lze předpokládat, že se může jednat o chybu Qt v kombinaci s mou konfigurací počítače. Pro správné zobrazování bylo nutné okno s náhledem skrýt a znovu zobrazit. Následně již překreslování fungovalo správně. U přepnutí do režimu celé obrazovky však tato chyba nastala 0 až 1 sekundu po přepnutí do tohoto režimu. Proto bylo nutné při přehrávání v režimu celé obrazovky počkat 1 sekundu po přepnutí a následně skrýt a zobrazit okno a teprve poté spustit animaci. Čekání jednu sekundu není pro uživatele nijak zatěžující a současně odstraníme nežádoucí chování. Po otestování na jiných počítačích k těmto

vadám nedocházelo. Proto se jedná nejspíše o chybu zaviněnou konfigurací počítače. Uvedené ošetření chyby bylo zachováno, protože nelze předvídat na kterých dalších konfiguracích by se mohla chyba také projevit.

Ovládání programu není v této práci popsáno. Návod na ovládání lze nalézt na přiloženém disku CD-ROM. Snaha byla vytvořit intuitivní uživatelské rozhraní, ale některá specifika tohoto nástroje vyžadují alespoň částečné nastudování příručky pro použití. Pro další vývoj by bylo nutné program otestovat na větším počtu uživatelů a dle jejich reakcí a připomínek upravit rozhraní, tak aby uživatelům vyhovovalo.

5 Výsledky

Výsledný program bylo nutné otestovat a zjistit, jakým způsobem je schopen běžet na různých počítačích. Testování bylo provedeno pomocí vytvoření testovacích animací a sledováním rychlosti přehrávání. Pokud pracuje program rychleji než je nutné, tak tuto skutečnost nepoznáme, protože k vykreslení scény dochází v pravidelných intervalech. Dle nastavení program zpracovává 25 nebo 50 snímků za sekundu. Jsme schopni pozorovat, zda nedochází ke zpomalení. To lze provést měřením délky přehrávání zadané sekvence a porovnáním délky přehrané sekvence a očekávané délky. Z těchto dvou hodnot zjistíme, jak dochází ke zpomalení v různých situacích.

Testování bylo provedeno na notebooku s operačním systémem Windows Vista. Notebook obsahuje procesor Intel Core 2 Duo s taktovací frekvenci 2,26GHz. Obsahuje grafický čip Intel Mobile 4 Series s velikostí grafické paměti 770MB. Pracuje v rozlišení 1280x800. Grafický výkon tohoto počítače není příliš velký, ale pro potřeby implementovaného editoru dostačoval. Grafický čip s dostupnými ovladači pracuje s verzí OpenGL 2.1. Podporuje tedy využití uvedených shaderů.

Nejprve byl otestován vliv počtu polygonů scény na rychlost vykreslování. Byla přehrána jedna scéna o délce sekvence 30s a postupně byl zvyšován počet polygonů, ze kterých se scéna skládá. Z tabulky 5.1 je patrné, že maximální použitelný počet polygonů je 128x128. Větší počet nepřináší vizuální vylepšení, ale pouze výkonovou zátěž. Počet snímků za sekundu nemá pro testovanou sestavu vliv. Je vhodnější použít 50 snímků za sekundu, protože je pohyb při animaci jemnější.

Počet vykreslovaných polygonů	Doba přehrávání (25 FPS)	Doba přehrávání (50 FPS)
0: 1x1 = 1 polygon	30s	30s
1: 2x2 = 4 polygony	30s	30s
2: 4x4 = 16 polygonů	30s	30s
3: 8x8 = 64 polygonů	30s	30s
4: 16x16 = 256 polygonů	30s	30s
5: 32x32 = 1 024 polygonů	30s	30s
6: 64x64 = 4 096 polygonů	30s	30s
7: 128x128 = 16 384 polygonů	30s	30,1s
8: 256x256 = 65 536 polygonů	70s	87s
9: 512x512 = 262 144 polygonů	177s	325s
10: 1024x1024 = 1 048 576 polygonů	643s	1276s

Tab. 5.1 Rychlost vykreslování scény

Pro otestování rychlosti při využití efektů byla vytvořena animace 10 scén, kdy každá trvá 3s. To dává dohromady sekvenci dlouhou 30s. Jako další parametry byl zvolen počet polygonů a to na

hodnotě kvality 7, která byla v předchozím případě poslední, kterou testovací sestava zvládala. Nejlépe se při ní projeví případné zpomalení. Byla zvolena snímkovací frekvence 50FPS. Nejprve byly v tabulce 5.2 otestovány obrazové efekty s využitím fragment shaderů. Implementované efekty nemají vliv na rychlost vykreslování. Zejména z toho důvodu, že jsou poměrně jednoduché. U složitějších efektů by ke zpomalení docházelo.

Typ efektu	Doba přehrávání
bez efektu	30,2s
negativ	30,2s
stupně šedi	30,25s

Tab. 5.2 Rychlost obrazových efektů

Dále byl otestován vliv přechodových efektů na rychlost vykreslování. Přechodové efekty jsou náročné z toho důvodu, že musí být vykresleny scény dvě. Pro otestování byla využita stejná scéna jako v předchozím případě. U každé scény byl nastaven přechod o délce 1s. Při přehrávání scény o délce 3s je tato doba rozdělena. Přechod z předchozí scény trvá 1s, následně je 1s přehrávána jediná scéna a další 1s trvá přechod na scénu následující. Celková doba přehrávání by tedy měla být 10*2s+1s=21s. Byl zvolen jeden přechod jako zástupce maskování a to přechod zleva doprava. Ostatní maskovací efekty by měly mít výsledky podobné.

Z tabulky je patrné zpomalení animace přibližně o 3 sekundy. To se děje z toho důvodu, že při přechodech musí být zobrazovány dvě scény současně. Rychlost při použití maskování je mírně rychlejší a to pravděpodobně z toho důvodu, že se překresluje pouze část scény. V případě prolnutí se musí vykreslit obě dvě scény celé. Pokud dochází k takovému zpomalení při přehrávání, lze rychlost přehrávání zvýšit snížením kvality zobrazení. Pro tento případ postačilo snížení kvality vykreslování scény na stupeň 6.

Typ efektu	Doba přehrávání
maskování	23,9s
prolnutí	24,1s

Tab. 5.2 Rychlost přechodových efektů

Přehrávání bylo otestováno také na starším počítači s operačním systémem Windows XP, procesorem mobile AMD Athlon XP+2700, 1,79GHz a grafickým čipem Radeon IGP-320M. Přesto, že se jedná o poměrně starý notebook, bylo možné animaci spustit bez výraznějšího trhání obrazu. Při přehrávání byl snížen počet snímků na 25 za sekundu a kvalita obrazu byla snížena na 5. Docházelo k drobnému zpoždění, avšak obraz byl poměrně plynulý. Delší byla pouze doba načítání projektu. Tato sestava nepodporuje OpenGL verze 2 a některé rozšíření. Proto na této sestavě nebyly funkční obrazové efekty a také prolínání obrazu. Tvorba animací by na tomto počítači nebyla tak plynulá jako na předchozí sestavě, ale přehrávání by nemělo činit žádné problémy.

Program byl také otestován na školním počítači v CVT. Byl spuštěn na počítači s operačním systémem Windows XP. Obsahoval procesor Intel Core 2 Duo s taktovací frekvenci 2,66GHz a grafickou kartou NVIDIA GeForce 7300 GT. Procesor v této sestavě je mírně rychlejší než procesor u testovací sestavy, ale grafický výkon tohoto počítače je výrazně vyšší. I proto bylo možné na této sestavě spouštět přehrávání až s kvalitou 8. Aplikace na této sestavě reagovala nejlépe ze všech testovaných sestav. Při spouštění nenastaly žádné problémy a všechny implementované funkce fungovaly správně.

6 Závěr

Podle zde uvedeného postupu byl vytvořen editor pro 3D projekci fotografií. Kromě základních požadovaných funkcí rotace a přiblížení byly implementovány další rozšíření. Nástroj umožňuje pracovat s kamerou pomocí zadaných řídících bodů animace nebo tyto body proložit křivkou, což zaručí hladší průběh při pohybu. Dále je možné měnit charakteristiku průběhu kamery a tím měnit její rychlost při pohybu scénou. Byly implementovány další rozšíření jako obrazové efekty, které dokáží s využitím fragment shaderů změnit charakter výsledného obrazu. Také byly přidány přechodové efekty, které umožní plynulý přechod z jedné scény do druhé. Pro editor bylo navrženo a vytvořeno uživatelské rozhraní. Výsledný nástroj umožňuje vytvářet zajímavé animace. Po seznámení s programem je tvorba těchto sekvencí poměrně jednoduchá.

Přínosem pro mne bylo nastudování funkce objektivu fotoaparátu. Před tvorbou této práce jsem měl pouze povrchní znalosti v této oblasti. Po dokončení této práce je mi bližší činnost fotoaparátů a vliv jeho vlastností na výsledné fotografie. Dále jsem se seznámil s knihovnou Qt a způsobem práce v ní. Knihovna mi dokázala ušetřit spoustu práce. Vyzkoušel jsem si tvorbu uživatelského rozhraní u aplikace, která ještě nemá jiné alternativy. Nemohl jsem tedy využít znalosti jiného rozhraní existující aplikace, ale musel jsem navrhnout nové rozhraní, které by se způsobem práce blížilo aplikacím, se kterými umí uživatelé pracovat a které znají. Tím, že podobná aplikace není k dispozici, nebylo možné vytvořené rozhraní porovnat s jiným.

V případě dalšího rozšiřování by bylo nutné přidat další přechodové a obrazové efekty. Také by bylo nutné přidat další průběhy kamery v závislosti na pozici. Ty vytvořené jsou pouze základní a kladou si za cíl demonstrovat způsob vytváření efektů dalších. Dále by bylo nutné otestovat uživatelské rozhraní na větším počtu uživatelů a sledovat jejich připomínky k rozhraní. Na základě připomínek pak současné uživatelské rozhraní upravit tak, aby lépe vyhovovalo přáním uživatelů. Při tvorbě programu byla snaha zaměřená na optimalizování přehrávání. Při dalším rozšiřování by bylo vhodné se zaměřit také na optimalizaci načítání projektů. Zajímavé by také mohlo byt rozšíření v podobě přidání zvukových efektů a hudby do animace a také export projektu do video-souboru.

Literatura

[1] Halliday, D., Resnick, R., Walker, J.: Fyzika - Vysokoškolská učebnice obecné fyziky. VUTIUM. Brno. 2000.

[2] Dvořák, D.: Digitální fotoaparát III: Optika. 5. března 2003. Dokument dostupný na URL http://www.digimanie.cz/art_doc-7F83734BA3AA7943C1256CC600313FF2.html (květen 2010)

[3] Tipy objektivů. DIFOTO. 23. července 2009. Dokument dostupný na URL http://www.difoto.cz/clanek.php?cis=11 (květen 2010)

[4] Dvořák, D.: Digitální fotoaparát IV: Vady zobrazení. 6. března 2003. Dokument dostupný na URL http://www.digimanie.cz/art_doc-FBF5337509C369C6C1256CC600319220.html (květen 2010)

[5] Hučka, R.: Vývoj a konstrukce fotografických objektivů.

[6] Žára, J., Beneš, B., Sochor, J., Felkel, P.: Moderní počítačová grafika. Computer Press. Brno. 2004

[7] Kršek, P.: Základy počítačové grafiky – studijní opora. Brno. 2006.

Příloha A: Příklad konfiguračního souboru

```
<?xml version="1.0" ?>
<anim>
    <head>
        <count value="2" />
        <project>Testovaci video</project>
        <author>Martin Dokoupil</author>
        <directory>Foto/</directory>
    </head>
    <scene id="1">
        <shead>
            <src>foto 011.bmp</src>
            <lenght value="300" />
            <lens focallenght="28.000000" type="0" />
            <spline x="0" y="0" z="0" fl="0" />
            <camerapath isx="0" isy="0" isz="0" isfl="0"
                        x="0" y="0" z="0" fl="0" />
            <effect lenght="200" type="0" fade="1" />
            <postproc type="0" />
        </shead>
        <move anglex="0" angley="0" anglez="0" zoom="68" time="0"
              funcx="0" funcy="0" funcz="0" funczoom="0" />
        <move anglex="-23" angley="-14" anglez="0" zoom="28"
              time="300" funcx="0" funcy="0" funcz="0"
              funczoom="0" />
    </scene>
    <scene id="2">
        <shead>
            <src>foto 012.bmp</src>
            <lenght value="300" />
            <lens focallenght="65.000000" type="0" />
            <spline x="0" y="0" z="0" fl="0" />
            <camerapath isx="0" isy="0" isz="0" isfl="0" x="0" y="0"
                        z="0" fl="0" />
```

```
            <effect lenght="100" type="2" fade="0" />
            <postproc type="1" />
        </shead>
        <move anglex="-12" angley="-3" anglez="0" zoom="48" time="0"
              funcx="0" funcy="0" funcz="0" funczoom="0" />
        <move anglex="0" angley="0" anglez="0" zoom="48" time="251"
              funcx="0" funcy="0" funcz="0" funczoom="0" />
        <move anglex="-12" angley="-3" anglez="0" zoom="28"
              time="300" funcx="0" funcy="0" funcz="0"
              funczoom="0" />
    </scene>
</anim>
```